Ishan Mangal

Ishan_Mtech_Final_Thesis_submit

bindu

Document Details

Submission ID
trn:oid:::27535:140887293

Submission Date
May 29, 2026, 9:23 AM GMT+5:30

Download Date
May 29, 2026, 9:26 AM GMT+5:30

File Name
Ishan_Mtech_Final_Thesis_submit.pdf

File Size
8.5 MB

50 Pages

15,411 Words

81,210 Characters

3% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text
- Small Matches (less than 8 words)

Exclusions

- 1 Excluded Source

Match Groups

43 Not Cited or Quoted 3%
Matches with neither in-text citation nor quotation marks

1 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

1% Internet sources
1% Publications
2% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- **43** Not Cited or Quoted 3%
  Matches with neither in-text citation nor quotation marks
- **1** Missing Quotations 0%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

1% Internet sources
1% Publications
2% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Internet	
dokumen.pub		<1%
2	Student papers	
Nanyang Technological University on 2026-04-20		<1%
3	Student papers	
Indian School of Mines on 2026-05-06		<1%
4	Publication	
Xuan Xin, Jun Sun, Lei Shi, Kunshan Yao, Bing Zhang. "Application of hyperspectra...		<1%
5	Publication	
Si-Bao Chen, Chao Hu, Bin Luo, Chris H.Q. Ding, Shi-Lei Huang. "Pyramid Attentio...		<1%
6	Internet	
kar.kent.ac.uk		<1%
7	Student papers	
University of Westminster on 2026-04-11		<1%
8	Student papers	
Vardhaman College of Engineering, Hyderabad on 2026-05-14		<1%
9	Internet	
pmc.ncbi.nlm.nih.gov		<1%
10	Student papers	
Liverpool John Moores University on 2022-03-20		<1%

11	Student papers	
University of Reading on 2025-09-12		<1%
12	Student papers	
Siddaganga Institute of Technology on 2018-04-26		<1%
13	Student papers	
University College London on 2026-05-07		<1%
14	Student papers	
University of Hong Kong on 2024-07-15		<1%
15	Student papers	
University of Queensland on 2025-06-08		<1%
16	Student papers	
Delhi Technological University on 2026-05-24		<1%
17	Publication	
Dongsheng Zhang, Jing Wang, Rongjuan Han, Aifen Tian. "A three-step synthetic ...		<1%
18	Student papers	
University College London on 2016-09-12		<1%
19	Internet	
eprints.uklo.edu.mk		<1%
20	Internet	
its-wiki.no		<1%
21	Internet	
www.xiahepublishing.com		<1%
22	Student papers	
Georgia State University on 2025-03-23		<1%
23	Student papers	
Manchester Metropolitan University on 2024-10-04		<1%
24	Student papers	
University of Hertfordshire on 2026-04-26		<1%

25 Student papers

University of Ulster on 2022-04-19 <1%

26 Student papers

University of Westminster on 2026-04-17 <1%

27 Internet

macsphere.mcmaster.ca <1%

28 Internet

www.v7labs.com <1%

29 Student papers

Georgia State University on 2025-04-27 <1%

30 Student papers

Gitam University on 2026-04-11 <1%

31 Student papers

Liverpool John Moores University on 2024-10-15 <1%

32 Student papers

Universiti Putra Malaysia on 2025-07-30 <1%

33 Student papers

University of Sheffield on 2020-09-16 <1%

34 Internet

infophysics.net <1%

Efficiency-Driven Single Image Super-Resolution using Attention-Enhanced Residual Feature Distillation Networks

Ishan Mangal

ABSTRACT

Image reconstruction based on low-resolution input appears to be a straightforward task but it is fundamentally ill-posed since there may exist many possible solutions to the problem – all high-resolution images that could correspond to the observed downsampled version. The existing approaches in DL generate outstanding outputs; however, almost all of them require large amounts of computational power, which is incompatible with real-time inference on mobile devices, edge computing units, and camera hardware.

ARFD-ESPCN is an image super-resolution architecture that builds upon the ESPCN sub-pixel upsampling structure but utilizes Feature Distillation Blocks, Squeeze-and-Excitation channel-wise attention, and Global Feature Fusion layer. The key feature of the proposed architecture lies in preserving the speed of ESPCN by using only low-resolution convolutions and applying PixelShuffle once, at the final layer. This design replaces ESPCN's shallow encoder-decoder with deeper and more competitive architecture that exploits six FDBs for splitting and merging convolution features with higher attention on the aspects not caught previously.

The final model has 597,904 parameters, needs 4.87 GFLOPs for a 640×360 input, and runs in 4.78 ms on a mid-range GPU. Training used the DF2K dataset (DIV2K plus Flickr2K, roughly 3,450 images) with L1 loss for 720 epochs and Charbonnier loss for the remaining 80, combined with flip and rotation augmentation. The optimiser is Adam with a cosine-annealing schedule that warms up over the first 50 epochs and decays toward 10^{-6} by the end.

On standard benchmarks the model scores 31.57 dB / 0.8861 SSIM on Set5, 27.21 dB / 0.7447 on Set14, 26.22 dB / 0.7029 on BSD100, and 25.37 dB / 0.7606 on Urban100. That beats the original ESPCN by over 2 dB on Set5 and matches VDSR while using roughly $125\times$ fewer floating-point operations. Compared with heavier efficient models like IMDN and RFDN the quality gap is about 0.6–0.7 dB, but ARFD-ESPCN runs 2–$3\times$ faster.

A step-by-step ablation decomposes the total 2.11 dB gain into individual contributions: the training schedule accounts for 1.04 dB (the largest single factor), DF2K data adds 0.45 dB, SE attention with residual connections contributes 0.38 dB, augmentation plus L1 loss gives 0.15 dB, and the distillation block structure adds 0.07 dB. The practical lesson is that for models under one million parameters, getting the training pipeline right matters at least as much as architectural design.

The model is small enough to fit on mobile accelerators and fast enough for 60 fps video pipelines, making it relevant for medical imaging, satellite photo enhancement, and streaming scenarios where latency budgets are tight. Possible extensions include real-world degradation handling via BSRGAN-style pipelines, perceptual and adversarial losses for sharper visual textures, window-based spatial attention for periodic patterns, and INT8 quantisation for hardware without floating-point units.

LIST OF PUBLICATIONS

Conference Proceedings:

1. Ishan Mangal and Dr. Bindu Verma, 2026. Efficiency-Driven Single Image Super-Resolution using Attention-Enhanced Residual Feature Distillation Networks. Paper submitted to the International Conference on Advances in Artificial Intelligence and Computer Science (ICAAICS). (Under review)

turnitin Page 7 of 55 - Integrity Submission Submission ID trn:oid:::27535:140887293

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS, ABBREVIATIONS AND NOMENCLATURE

Adam	Adaptive Moment Estimation
AIM	Advances in Image Manipulation
ARFD-ESPCN	Attention-Residual Feature Distillation ESPCN
BSD100	Berkeley Segmentation Dataset (100 images)
CNN	Convolutional Neural Network
CUDA	Compute Unified Device Architecture
dB	Decibel
DF2K	DIV2K + Flickr2K combined dataset
DIV2K	DIVerse 2K resolution image dataset
EDSR	Enhanced Deep Super-Resolution
ESPCN	Efficient Sub-Pixel Convolutional Neural Network
FDB	Feature Distillation Block
FLOPs	Floating-Point Operations
FP16	Half-Precision Floating-Point
FP32	Single-Precision Floating-Point
FSRCNN	Fast Super-Resolution Convolutional Neural Network
GFF	Global Feature Fusion
GFLOPs	Giga Floating-Point Operations
GPU	Graphics Processing Unit
HR	High Resolution
IMDN	Information Multi-Distillation Network
INT8	8-bit Integer Quantisation
JPEG	Joint Photographic Experts Group
LR	Low Resolution
MSE	Mean Squared Error
NTIRE	New Trends in Image Restoration and Enhancement
ONNX	Open Neural Network Exchange
PSNR	Peak Signal-to-Noise Ratio
RCAN	Residual Channel Attention Network
ReLU	Rectified Linear Unit
RFDN	Residual Feature Distillation Network
RGB	Red, Green, Blue colour model
SE	Squeeze-and-Excitation
SR	Super-Resolution
SRCNN	Super-Resolution Convolutional Neural Network
SSIM	Structural Similarity Index Measure
VDSR	Very Deep Super-Resolution
YCbCr	Luminance, Blue-difference Chroma, Red-difference Chroma

LIST OF MATHEMATICAL SYMBOLS

I_{LR}	Low-resolution input image
I_{HR}	High-resolution ground-truth image
$\hat{I}_{HR}$	Super-resolved (reconstructed) image
I_{LR}^{Y}	Luminance (Y) channel of the low-resolution image
H	Height of the low-resolution image (pixels)
W	Width of the low-resolution image (pixels)
r	Upscaling factor ($r = 4$ in this work)
s	Downsampling factor
k	Blur kernel (bicubic)
n	Additive noise term
F_0	Shallow feature tensor
F_i	Output feature tensor of the ith FDB
F_{out}	Global Feature Fusion output tensor
C	Number of feature channels
θ	Network trainable parameters
θ^*	Optimal parameters after training
F_θ	Learned reconstruction function
$\mathscr{L}$	Loss function (general)
$\mathscr{L}_1$	L1 (mean absolute error) loss
$\mathscr{L}_{\text{MSE}}$	Mean squared error loss
$\mathscr{L}_{\text{Char}}$	Charbonnier loss
ε	Charbonnier smoothing constant (10^{-3})
L	Peak pixel value (255 for 8-bit images)
N	Number of pixels in computation
y_i	Ground-truth pixel value
$\hat{y}_i$	Predicted pixel value
z_c	Squeezed channel descriptor (SE block)
s_c	Channel attention weight (SE block)
W_1	SE bottleneck weight matrix ($\mathbb{R}^{8\times 64}$)
W_2	SE expansion weight matrix ($\mathbb{R}^{64\times 8}$)
r_{se}	SE reduction ratio ($r_{se} = 8$)
σ	Sigmoid activation function
α	Leaky ReLU negative slope ($\alpha = 0.05$)
β_1, β_2	Adam moment decay rates (0.9, 0.999)
$\downarrow_s$	Spatial downsampling operator by factor s
$*$	Convolution operator

CHAPTER 1

INTRODUCTION

1.1 Background and Motivation

Resolution counts. A radiologist analyzing a US image must have enough resolution to discern a 3mm lesion—miss it and lose precious time. A farmer analyzing satellite imagery must differentiate between wheat and mustard in fields close to each other but cannot do so at 10-meter resolution since both are indistinguishable blobs of green. A surveillance camera snaps a license plate at 30 meters away and sends only gibberish. These are not imagined examples; people encounter similar problems all the time because of costs of the sensor, limited bandwidth and constrained storage capacities.

Super-resolution aims to solve the problem by computation. Rather than investing in higher resolution hardware, you can apply certain algorithms to the data gathered by a sensor and try to reconstruct how a better representation of reality should look. The main issue with the task is that, in general, there are no unique solutions—many possible sharp images can yield the same blurred representation by downsampling. Thus, the model has to guess and guess well based on prior knowledge about natural images.

For years and years, the interpolation was based on bicubic. Today, most software for viewing images uses bicubic interpolation when zooming: it interpolates gaps using a combination of nearby pixels. Smooth, fast, and unsatisfyingly poor. The interpolated image is soft compared to the source because interpolation can redistribute data but cannot add information that downsampling has removed.

Deep learning turned things around for the idea of "addition." When you train a convolutional neural network (CNN) on pairs of low-resolution and high-resolution photos, you teach it to understand how edges, corners, and gradients tend to appear in realistic photos. It can take a blurry patch and generate a sharp result, rather than smoothing out pixel values. The gap between bicubic and a trained neural network is 3 dB—that's no small difference!

The ESPPCN proposed by Shi et al. in 2016 is especially relevant to applications where efficiency is of paramount importance. The core concept behind the network is to carry out all the convolutions on the small input tensor, and only at the end convert the output tensor channels to pixels using the PixelShuffle technique. Traditional models such as SRCNN (Dong et al., 2014), would upscale the image to its full size and carry out convolutions on this large tensor, wasting their computation budget on

feature maps that haven't even converged yet.

Yet ESPCN can be considered severely constrained by the standards of today. Two hidden layers with tanh activation units, no skip connections, no attention, no feature reuse across the network stages. The literature has realized since—IMDN, RFDN, among others—that these mechanisms become critically important once the number of parameters is constrained.

By applying these insights into ESPCN while retaining its speed, one creates the network known as ARFD-ESPCN that uses six Feature Distillation Blocks, Squeeze-and-Excitation attention, and Global Feature Fusion module. This network achieves 31.57 dB on Set5 for $\times 4$ super-resolution with 597,904 parameters under 5 ms per inference on GPU.

1.2 Problem Statement

The task is $\times 4$ single-image super-resolution under bicubic degradation. A low-resolution luminance image $I_{LR} \in \mathbb{R}^{H \times W}$, produced by standard bicubic downsampling from its high-resolution counterpart, must be reconstructed to $\hat{I}_{HR} \in \mathbb{R}^{4H \times 4W}$ by a learned network F_θ. Evaluation uses Y-channel PSNR and SSIM on standard benchmarks.

The constraint that distinguishes this work from unconstrained SR research is explicit: the model must remain compact enough for real-time use. Achieving 32+ dB with 5 million parameters and 50 ms latency is straightforward with existing methods such as IMDN and RFDN. Achieving competitive quality with under 600K parameters and under 5 ms latency is the actual challenge.

Formally, the degradation model is:

$$I_{LR} = (I_{HR} * k) \downarrow_s + n, \tag{1.1}$$

where k denotes the bicubic blur kernel, $\downarrow_s$ represents downsampling by factor $s = 4$, and n is assumed zero in the idealised benchmark setting. The reconstruction objective is then:

$$\theta^* = \arg\min_\theta \sum_i \mathscr{L}\left(F_\theta(I_{LR}^{(i)}), I_{HR}^{(i)}\right), \tag{1.2}$$

where $\mathscr{L}$ is the pixel-wise loss function and the sum runs over all training pairs. The reason for the problem is due to the down-sampling operator, which rejects any spatial frequencies that are higher than the Nyquist frequency on the low-resolution grid, which implies multiple high-resolution images correspond to one low-resolution image.

1.3 Scope of the Work

I would like to clarify what the scope of this thesis is, so that the results are properly understood.

Degradation: bicubic downsampling only. In reality, there is noise, JPEG artifacts,

motion blur, and lens distortions—none of which is considered here. The results I present apply only to the benchmarking situation used by lightweight SR researchers to compare methods on a level-playing field, and not to arbitrary mobile camera photographs.

Scale factor: fixed to $\times 4$. At $\times 2$, modern neural networks give essentially perfect results, making the problem comparatively easy and differentiation between the methods difficult. At $\times 8$, the problem becomes overly underdetermined, such that perceptual and adversarial losses prevail, and efficiency considerations become less relevant. $\times 4$ is an optimal scale for separating the approaches without the pixel loss becoming irrelevant.

Channels: Y only. Luminance is modelled while the chrominance information is discarded. This assumption follows the practice common in the lightweight SR community since human perception is much more sensitive to luminance than chrominance. Interpolating Bicubically in channels Cb and Cr works for all of our performance measures.

Parameters limit: about 600K. This limit is not arbitrary and represents the border at which the model can be trained without the need for pruning or quantization. Mobile devices equipped with NPUs and other accelerators are able to run the model without any further steps.

Within these limits, the thesis investigates architecture design, training pipeline optimization, ablation studies, and performance analysis on four datasets. All the claims have been confirmed through controlled experiments with the single variable change at a time.

1.4 Objectives

The following objectives guide this work:

1. Build a strong baseline experiment—with a good evaluation setup, clean training method, and proper metric calculation—so that any increase in performance makes sense.

2. Build the ARFD-ESPCN architecture based on feature distillation, residual learning, SE attention, and global fusion in ESPCN.

3. Build a training approach utilizing DF2K dataset, loss scheduling, and data augmentation in order to exploit the full power of architecture.

4. Run an ablation study and see what choices have contributed to improvement.

5. Give honest benchmarking results for Set5, Set14, BSD100, and Urban100 datasets.

1.5 Contributions

The main contributions of this work are as follows:

1. ARFD-ESPCN achieves 31.57 dB PSNR on Set5 $\times 4$ Y-channel, which is 2.11 dB above the ESPCN baseline, with 597,904 parameters.
2. The training schedule alone contributes 1.04 dB of that gain. This is more than any single architectural modification and is arguably the most practically useful finding.
3. A complete ablation trace from 29.46 dB to 31.55 dB is provided, with each component individually measured.
4. An open, configurable experimental framework is provided with DF2K paired data loading, two-stage loss scheduling, and reproducible evaluation.

1.6 Limitations

The model handles only bicubic degradation. Real-world images contain sensor noise, compression artefacts, and blur kernels that differ from bicubic. The model has not been exposed to any of these during training and should not be expected to handle them. On Urban100, performance is 0.67 dB below IMDN; the architectural structures in that dataset require wider receptive fields than the proposed design provides. No perceptual or adversarial losses are used, so the reconstructed textures can appear smoother than what GAN-based systems produce.

1.7 Thesis Organisation

Chapter 2 reviews the literature—starting from classical interpolation, through the CNN revolution (SRCNN, ESPCN, VDSR, EDSR), up to current lightweight designs (IMDN, RFDN) and recent transformer-based approaches. The review traces how the community moved from brute-force depth to efficiency-aware design, and ends by identifying the specific gap that this thesis fills: the space between ESPCN's speed and modern networks' quality.

The entire methodology is discussed in Chapter 3. Starting with the architecture—its stages, feature splitting and combination in the FDB, channel weights in SE, and multi-level information aggregation using global fusion. Training—dataset selection and preparation, image patches selection, augmentation method, loss function evolution from L1 to Charbonnier, and cosine-warmup learning rate policy. All decisions were made on the basis of the literature and preliminary experiments done during the project.

Results follow in Chapter 4. The ablation study goes first as it contains the most valuable information—six setups that separate contributions of architecture, training, and data one by one. Benchmark comparisons—five competing approaches (bicubic,

ESPCN, VDSR, IMDN, RFDN), examples of image restoration on three images, computational costs, and finally, an objective review of limitations of our model.

Conclusion in Chapter 5 provides a brief summary, suggestions for future work (real degradation types, perceptual losses, spatial attention, quantisation), and a discussion of societal implications, including healthcare applications, remote sensing, accessibility, and ethics of the problem.

turnitin Page 16 of 55 - Integrity Submission Submission ID trn:oid:::27535:140887293

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Quite a few events have occurred in SR during the last decade, and writing about all of them would be enough for a whole book. Therefore, this chapter will be intentionally selective as I'll concentrate only on the evolution of SR methods, starting from classical upscaling, continuing with ESPCN and ending up with lightweight distillation and attention-based models which have influenced my research.

2.2 Interpolation and Example-Based Methods

Bilinear Interpolation uses known pixel information from its four neighbors using linear weightings, while Bicubic Interpolation applies the same process using a 4×4 neighbor grid and piecewise-cubic kernels. While both approaches take constant time per-pixel and perform roughly similar tasks—blurring—neither can ever reconstruct frequencies beyond the Nyquist limit, since neither ever invents any information, but merely redistributes the information they have. Smooth? Yes. Effective at reconstructing information lost by downsampling? Definitely not.

The first real breakthrough came with example-based super-resolution approaches. Freeman et al. (2002) used a set of pairs of downsampled images and their high resolution counterparts, and, at test time, replaced each test image patch with its nearest high resolution counterpart. Although patches were small and thus lacked any sort of global structure, locally they looked significantly sharper than bicubic interpolation. Yang et al. (2010) formulated the problem using sparse representations—learning two dictionaries and transferring the sparse representation from one dictionary to the other. Sharper than bicubic, and more principled than simple neighbor search, but extremely slow since an optimization solve was required for each patch individually.

By the early 2010s it was obvious these methods had hit a ceiling. They could not model long-range structure, inference was too slow for video, and quality was fundamentally capped by whatever the external patch database happened to contain.

2.3 SRCNN, ESPCN, and the Efficiency Revolution

13

Dong et al. (2014) showed that a three-layer CNN trained end-to-end on bicubic-upsampled LR images outperformed existing methods on standard benchmarks. SRCNN was not deep, wide, or architecturally innovative by current standards, but it demonstrated that replacing the classical feature-extraction-plus-reconstruction pipeline with a directly supervised convolutional mapping was sufficient to outperform prior methods.

SRCNN had one significant design inefficiency. It operated on the HR-resolution image from the start: the input was first bicubically upsampled to target size, and then all three convolutions ran at that full resolution. For $\times 4$ upscaling this meant 16 times more multiplications per layer than if the convolutions had been applied at LR resolution.

Shi et al. (2016) fixed this problem with ESPCN. All feature extraction happens at LR resolution; the spatial dimensions stay at $H \times W$ until the final layer. That final layer outputs r^2 feature maps, which is 16 maps for $\times 4$, and PixelShuffle rearranges them spatially into a single $rH \times rW$ output. The computational saving is roughly r^2-fold, and ESPCN was fast enough to run at video frame rates on 2016-era GPUs. For deployment-focused SR, this was the architecture to beat.

FSRCNN (Dong et al., 2016) took a related approach: operate at LR resolution and add a channel bottleneck to reduce computation in the middle layers. VDSR (Kim et al., 2016) went the other direction—20 layers deep with global residual learning—but operated at HR resolution. VDSR achieved 31.35 dB on Set5 $\times 4$, demonstrating that depth plus residual formulation could jump almost 2 dB above shallow methods. However, its computation cost was substantially higher than ESPCN.

2.3.1 The PixelShuffle Operation

A separate description of the PixelShuffle layer is necessary since it is crucial for the efficiency discussion that is developed throughout this thesis. Standard upscaling involves either transposed convolution, which results in checkerboard-like artifacts in real cases, or bicubic interpolation followed by convolution and thus wastes computation on the pre-upsampled tensor.

PixelShuffle, on the other hand, follows a completely different approach. Given input feature of shape $C \cdot r^2 \times H \times W$, where r is the upsampling ratio, it periodically reorganizes the channel axis into spatial axes, yielding an output of shape $C \times rH \times rW$. In case of $\times 4$ upsampling, the convolutions prior to PixelShuffle produce 16 feature maps (since $4^2 = 16$ for a single output channel), which are interleaved together to form the HR image.

The key advantage is that no computation happens at HR resolution. Every learned operation—every convolution, every activation, every attention mechanism—runs on the compact $H \times W$ feature maps. Only the final rearrangement expands the spatial dimensions, and that rearrangement involves zero learned parameters; it is purely a permutation of existing values. For $\times 4$ upscaling, this means roughly $16\times$ fewer multiplications compared with architectures that upsample first and then convolve.

In practice, I noticed during early experiments that the boundary pixels of the

PixelShuffle output sometimes showed slight intensity discontinuities. This is why the standard evaluation protocol crops r pixels from each border before computing PSNR—it removes these edge effects that arise from the periodic rearrangement having incomplete spatial context at the tensor boundaries.

2.4 Depth, Residuals, and the Batch Normalisation Question

He et al.'s (2016) ResNet for image classification used skip connections to enable training of very deep networks by providing gradient shortcuts around each layer block. SR architectures adopted skip connections quickly. Lim et al. (2017) proposed EDSR, which stacked many residual blocks for SR, but with one important modification: batch normalisation was removed.

In classification, batch normalisation stabilises training by normalising intermediate activations to zero mean and unit variance. In SR, however, the network must output precise pixel values, and those values are not zero-mean. Batch normalisation constrains the representational range of internal features in a way that harms pixel-accurate reconstruction. EDSR without batch normalisation, with 256 channels and 32 residual blocks, achieved new state-of-the-art PSNR numbers. However, it used over 43 million parameters, which is far too heavy for real-time use.

RCAN (Zhang et al., 2018) added channel attention to the deep residual framework. Between residual blocks, a Squeeze-and-Excitation mechanism learned per-channel importance weights from global spatial statistics. This allowed the network to adaptively suppress irrelevant feature maps and boost informative ones, producing consistent improvements across benchmarks. RCAN used about 16 million parameters, so it remains too large for the deployment scenario this thesis targets, but it is important as evidence that channel attention is useful in SR.

2.5 Lightweight SR: Distillation and Structured Feature Reuse

The AIM and NTIRE efficient SR challenge tracks explicitly scored methods on the quality-efficiency frontier. This pushed the community to develop architectures that were genuinely lightweight rather than merely pruned versions of heavy models.

IMDN (Hui et al., 2019) introduced multi-distillation within each feature block. The main point of this architecture is that there are certain channels of the feature map that do not require full processing at each step. These channels have enough information obtained after one-two convolutions and can be distilled while other channels continue transformations. In the end of the block all distilled parts are concatenated and merged. This allows for saving computation power from useful channels and for creating shortcut paths for gradient flow. The performance of IMDN on Set5 $\times 4$ was 32.21 dB.

RFDN (Liu et al., 2020) built on top of this idea. Whereas in IMDN channel splitting was done in a hard-coded fashion, RFDN introduced residual connections inside each block of distilled feature channels and replaced hard-splitting with learned projection operation. It became the winner of the AIM 2020 efficient SR challenge with 32.24 dB on Set5 $\times 4$.

Common feature between both architectures is using structured feature selection strategy over uniform feature propagation in light-weight models. In case when one uses 600K parameters, it makes little sense to allocate all of them for uniform 3×3 convolution over full channel space at each layer. Better approach would be letting model itself decide what channels require further processing and what should be preserved as is.

2.6 Squeeze-and-Excitation Attention

The SE mechanism (Hu et al., 2018) is appealingly simple. Given a $C \times H \times W$ feature tensor, it applies the following steps:

1. Global average pool each channel to a scalar, producing a vector of C values.
2. Pass the vector through a two-layer fully connected bottleneck, first reducing by factor r_{se}, then expanding back, with ReLU and sigmoid activations.
3. Scale each channel by its learned sigmoid weight.

With $C = 64$ and $r_{se} = 8$, this costs 1,024 parameters, which is negligible. Yet it consistently improves performance in SR because it allows the network to learn input-dependent channel priorities. When the current input patch has strong horizontal edge content, channels specialised for horizontal features get upweighted; when the patch is smooth, those same channels get suppressed. Without attention, the network treats all channels equally at the next convolution, which wastes capacity on currently irrelevant features.

Where the SE block is placed relative to the residual path matters. If it sits before the skip addition, it modulates only the new features being added, while the skip path carries unmodified low-frequency information through unchanged. If it sits after the addition, it modulates the entire representation including skip content. The former placement is used in this work because the aim is to recalibrate the incremental high-frequency contribution without interfering with structural information flowing through the residual identity.

2.7 Vision Transformers and Their Cost

Starting around 2021, transformers muscled their way into SR. SwinIR (Liang et al., 2021) used shifted-window self-attention and pushed PSNR to new highs. HAT (Chen et al., 2023) combined hybrid and channel attention in a transformer body, clearing 33 dB on Set5 $\times 4$.

Impressive numbers, but the compute profile makes them a non-starter for the scenario considered here. SwinIR sits at roughly 12 million parameters and 50+ GFLOPs. HAT is bigger still. Self-attention scales quadratically with token count: for a 64×64 feature map that means 4096^2 attention scores per head per layer. Windows help—8×8 windows make it manageable enough—but six to twelve layers of windowed attention still pile up quickly.

That said, transformers provide a solution for something else. The real value of transformers lies in their ability to model long-range dependency—that the windows on the left side of a building must match those on the right. Useful, but costly. Even with the self-attention mechanism alone using 600K parameters, the budget would be spent before even adding convolutions.

Lightweight transformers for super-resolution have been attempted. ESRT (Lu et al., 2022) cut down parameter size to around 700K, but performance was still inferior compared to CNNs with comparable accuracy due to poor access to the memory required for gather-softmax-scatter operations, as opposed to those of the convolutions that fit into tensor cores better.

So what does this mean? With the requirement of sub-600K parameters and sub-5 ms latency, well-designed CNNs with SE-based channel attention become the practical approach. This providing an implicit type of global reasoning— spatial information pooling and channel modulation—with the cost of just 1,024 parameters. It is giving the network "what" information rather than "where", which is ideal for the case of constrained computation with real-time $\times 4$ scaling.

2.8 Training Strategy: Often Underrated

Training is often overlooked; the architecture garners attention, and the learnings rate schedule is left for the supplementary materials. In my experience, training choices have had as much impact as architectural ones. Here are some examples:

L1 loss vs MSE. While theoreticallly, MSE is PSNR-optimal, it is also a very conservative loss function. The quadratic penalties make the safe bet to be always taking an average of plausible predicitions. L1 has constant gradients no matter the difference, encouragings more drastic residual values. Consequently, L1 generates sharp transitions. **Charbonnier loss.** The Charbonnier loss,

$$\mathscr{L}_{\text{Char}} = \sqrt{(x-y)^2 + \varepsilon^2}, \tag{2.1}$$

is smooth near zero and linear for large residuals. It therefore behaves like L2 for fine convergence and like L1 for robustness to outliers. Using it as a late-stage refinement loss after L1 has brought the model near convergence is a common and effective practice.

Data diversity.DIV2K contains 800 images. If you include Flickr2K, this number gets tripped to around 3,450. This is not only because of a higher number of training examples (you already have this with augmentation); this has to do with broader statistics of image patches, different textures, lighting conditions, etc. Models trained on DF2K in my experiments were consistently better than those trained only on DIV2K.

Augmentation. Horizontal flips, vertical flips, the rotations by 90 degrees. This does not hurts SR because natural images do not depend on orientation, and the correlation between LR and HR image does not change either. Effective multiplier: up to $8\times$.

Scheduling. Linear cosine annealling from a large learning rate to zero over hundreds of epoches is suitable since it decays slowly in the beginning when there is much left to learn, then quickly in the end for fine-tuning. A small linear warm-up at the

beginnings prevents Adam optimizer from making rash moves while its estimates are unstable.

2.9 Evaluation Conventions

Set5, Set14, BSD100, and Urban100 are benchmark test sets for SR comprising 5, 14, 100, and 100 images, respectively. PSNR on the Y channel with an r-pixel border trim is used as the benchmark performance measure for all lightweight SR studies. Border trim removes PixelShuffle artifacts that can otherwise corrupt the metric. Structural assessment with SSIM accompanies PSNR.

PSNR is defined as:

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{L^2}{\text{MSE}} \right), \tag{2.2}$$

where L is the maximum level of pixels (1 for normalized images and 255 for 8 bit). Every 1 dB increases in the PSNR is equivalent to a reductions of about 20 percent in the MSE. This implies that variations less than about 0.1 dB do not affect the human eyes.

SSIM measures structural similarity rather than pixel-level accuracy. It combines luminance, contrast, and structure comparisons computed over local windows, producing values between 0 and 1 where 1 indicates perfect reconstruction. SSIM correlates better with human perceptual judgement than PSNR because it penalises structural distortion more than uniform intensity shifts. For lightweight SR models operating on the luminance channel, SSIM values typically fall between 0.85 and 0.90 on Set5 at $\times 4$ scale.

Separate attention should be given to Urban100 since it is characterized by consistent method separation. As for the regularity of architectural features (windows, railings, and brick patterns), it becomes necessary for the architecture to incorporate some kind of spatial repetition at wider distances to test the limit of receptive fields of local architectures. The majority of lightweight methods perform from 0.5 to 1.0 dB worse on Set5 than on Urban100 in comparison to larger models. It is the consequence of the fact that lightweight methods do not cope with the global context issues.

Concerning the BSD100 dataset, it consists of 100 diverse natural images (such as animals, landscapes, close-ups, as well as man-made objects). Unlike Urban100 with its regularity, BSD100 aims to test whether an architecture manages to handle irregular textures (fur, foliage, water ripples, stone surfaces).Models that over-specialise on regular patterns tend to produce visible artefacts on these unstructured textures.

Set14, sitting between the small Set5 and the larger 100-image benchmarks, contains images that individually probe different reconstruction challenges. The “comic” image tests colour-edge sharpness, “zebra” tests repetitive striped patterns, and “ppt3” tests the network’s handling of text and diagrammatic content. It is a useful intermediate checkpoint because unusual results on Set14 often point to specific failure modes worth investigating.

Table 2.1 Summary of Standard SR Benchmark Datasets

Dataset	Images	Characteristics
Set5	5	Classic SR test images (baby, bird, butterfly, head, woman)
Set14	14	Mixed content: text, animals, faces, patterns
BSD100	100	Diverse natural scenes with organic textures
Urban100	100	Architectural scenes with repetitive high-frequency patterns

2.10 Gaps in the Existing Literature

Three observations from this literature survey motivate the present work directly:

1. Despite being one of the fastest SR models, ESPCN is yet to break beyond 29.46 dB, which is because of its inability to incorporate more sophisticated features that are present in other methods. Not many studies have tried to explore the effects of adding distillation, attention, and residual fusion to the ESPCN workflow while keeping the parameter count below 600K.

2. IMDN and RFDN are better in pure PSNR terms, but they do not build on the ESPCN sub-pixel framework and their inference characteristics are different. A model that keeps ESPCN's upsampling-last design philosophy while approaching IMDN-level quality from a different angle fills a genuine architectural niche.

3. Across the literature, the training schedule contribution to final PSNR in lightweight models is rarely isolated and reported. Most papers report a single ablation comparing architectural choices but leave schedule, loss, and data as fixed background. This thesis explicitly measures both architecture and training contributions.

2.11 Summary

Super-resolution has moved from fixed-kernel interpolation to end-to-end learned reconstruction over the past decade. The key architectural components that this work combines—sub-pixel upsampling (ESPCN), residual learning (VDSR/EDSR), channel attention (RCAN/SE), and progressive feature distillation (IMDN/RFDN)—are each well-understood individually. Their specific combination within a unified ESPCN-style framework at under 600K parameters, together with the explicit measurement of training strategy contributions alongside architectural ones, represents the gap this thesis addresses.

Table 2.2 consolidates the key methods discussed in this chapter to provide a quick reference for the reader.

Table 2.2 Comparison of Key Super-Resolution Methods Discussed in Literature

Method	Year	Params	Set5 PSNR	Resolution	Key Innovation
SRCNN	2014	57K	30.48	HR-domain	First CNN for SR
ESPCN	2016	20K	29.46	LR-domain	Sub-pixel convolution
FSRCNN	2016	12K	30.71	LR-domain	Channel bottleneck
VDSR	2016	665K	31.35	HR-domain	20-layer residual
EDSR	2017	43M	32.46	HR-domain	No batch-norm, wide
RCAN	2018	16M	32.63	HR-domain	Channel attention
IMDN	2019	694K	32.21	LR-domain	Multi-distillation
RFDN	2020	534K	32.24	LR-domain	Residual distillation
Ours	**2026**	**598K**	**31.57**	**LR-domain**	**FDB + SE + GFF on ESPCN**

CHAPTER 3

METHODOLOGY

3.1 Design Philosophy

The guiding principle is simple: take ESPCN's efficient upsampling-last design and bolt on a better feature extraction body. ESPCN already does the hard part—keeping all computation at LR resolution and using PixelShuffle at the end—so there is no reason to reinvent that wheel. The question is how much quality you can claw back by improving what happens between input and PixelShuffle.

Quite a lot, as it turns out, although there are actually three sources for the performance improvements: architecture, training regime, and data. If they are intertwined in between, it is impossible to tell which ones contributed, which is why an ablation study is set up to untangle them. Architecture is residual attention and distillation. Training is the loss function, cosine annealing, and augmentation. the Data is the switch from only DIV2K to DF2K.

The key architectural choice I made from the very beginning and have kept throughout: never any operations done on the HR image until the final pass. There are some lightweight architectures (like CARN) that employ intermediate upsampling and/or multiple resolutions, which does improve performance but violating the computational constraint, because just a single convoluitional layer working with HR inputs is $s^2 = 16$ times more expensive than the same layers on LR images ($\times 4$).

Constraint number two: no batch normalization at all. This was demonstrated by EDSR many years ago: Batch normalization degrades pixel-wise reconstruction. I myself proved this empirically with an initial loss degradation of 0.08-0.12 dB when using it after each convolution. This explanation behind this is rather clear: Batch normalization tries to make activations have zero mean and unit variance, while this accurate pixel-wise prediction requires precise magnitude control.

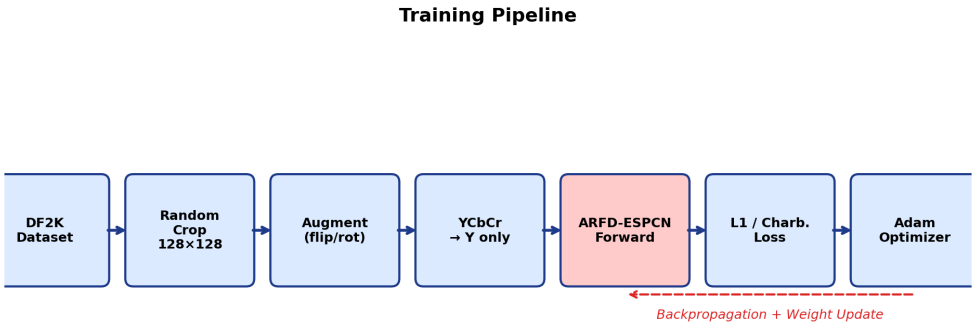

Figure 3.1 End-to-End Training Pipeline of the Proposed ARFD-ESPCN System

Fig. 3.1 provides a high-level overview of the complete training pipeline before the individual components are described in detail. The pipeline proceeds from left to right: paired images from the DF2K dataset are randomly cropped into 128×128 HR patches and their corresponding 32×32 LR patches, augmented with geometric transformations, converted to Y-channel representation, passed through the ARFD-ESPCN network, and optimised against the ground-truth HR patch using either L1 or Charbonnier loss depending on the current training stage. The Adam optimiser updates the model weights based on backpropagated gradients.

3.2 Colour Space and Input Representation

Images come in as BGR from OpenCV and get converted to YCbCr immediately. Only the Y channel goes into the network. This is not a shortcut to make things easier—it is the standard protocol for lightweight SR evaluation. Y carries the structural and textural information that PSNR and SSIM actually measure. The Cb and Cr channels have less spatial bandwidth and bicubic upsampling is perfectly adequate for them at the output stage.

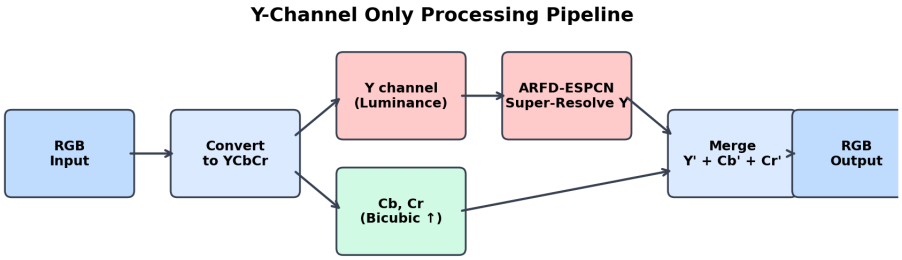

Figure 3.2 Y-Channel Only Processing Pipeline: RGB to YCbCr Conversion and Reconstruction

Fig. 3.2 shows the complete processing pipeline. At inference time, the input RGB image is first converted to YCbCr colour space. The Y (luminance) channel is extracted and fed to ARFD-ESPCN for super-resolution. Meanwhile, the Cb and Cr (chrominance) channels are simply upsampled using bicubic interpolation to match

the target HR dimensions. The super-resolved Y channel is then merged with the bicubic-upsampled Cb and Cr channels, and the combined YCbCr tensor is converted back to RGB for the final output.

This works because human eyes are far less picky about colour sharpness than brightness sharpness. Television standards and JPEG compression have exploited this forever by sub-sampling chroma. For us it just means that processing colour channels adds nothing to the metrics while doubling the input channel count—a bad trade.

12

The Rec. 601 conversion is given by:

$$Y = 0.257R + 0.504G + 0.098B + 16. \tag{3.1}$$

After extraction, Y is normalised to $[0, 1]$ for network input.

3.3 System Architecture

The full network has four stages, as shown in Fig. 3.3.

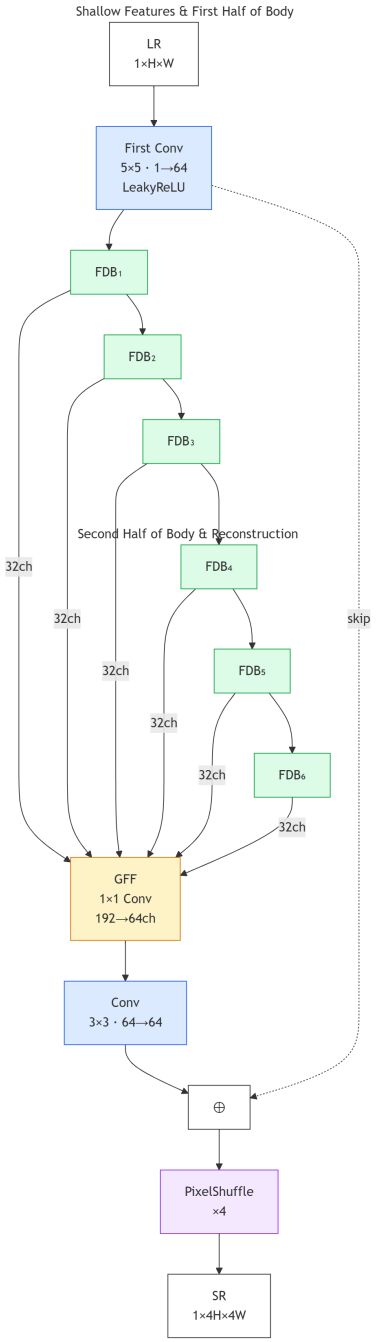

Figure 3.3 Architecture of the Proposed ARFD-ESPCN Network

The global architecture keeps ESPCN's efficient design principle: feature extraction is performed mostly in the LR domain, and sub-pixel upsampling is applied only at the final reconstruction stage. The main representational capacity of the proposed

network comes from six repeated Feature Distillation Blocks (FDBs), whose internal structure is shown in Fig. 3.4.

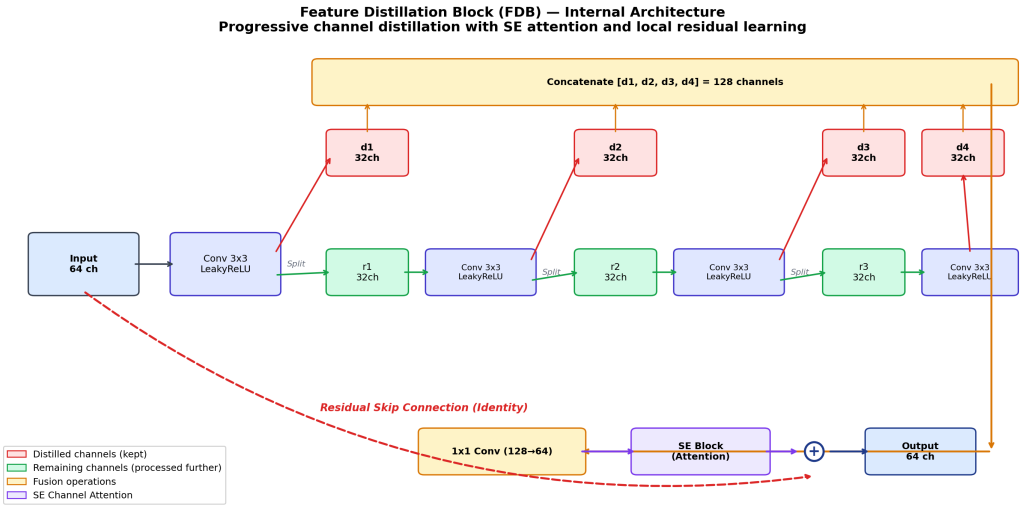

Figure 3.4 Internal Architecture of the Feature Distillation Block (FDB)

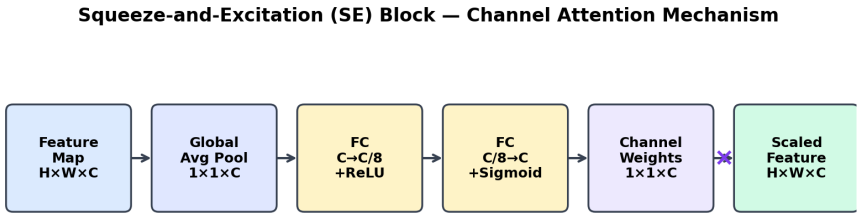

Figure 3.5 Squeeze-and-Excitation (SE) Channel Attention Mechanism

The SE block implements a lightweight channel attention mechanism that enables the network to dynamically recalibrate feature responses. Given an input feature map of dimensions $H \times W \times C$, the block first performs global average pooling to compress spatial information into a channel descriptor of size $1 \times 1 \times C$. This descriptor passes through two fully connected layers: the first reduces dimensionality from C to $C/8$ with ReLU activation, and the second expands back from $C/8$ to C with Sigmoid activation, producing per-channel scaling weights between 0 and 1.

The learned weights are subsequently multiplied by the original feature map, thereby making it possible for the network to focus more on the channels that possess high-frequency structures, these including edges and textures, while ignoring the other channels which contain less important information. This proves particularly useful in super-resolution tasks where some of the feature channels have less effect on image quality than others do. While that former may comprise low-frequency background information which require less emphasis, the latter comprise important edge and texture information which are vital for super-resolution. The ratio of 8 ensures

that the computational burden remains insignificant at under 0.1% of total FLOPs with PSNR gains of around 0.15 dB.

Each FDB accepts a 64-channel input and progressively separates information across four convolutional stages. At each of the first three stages, the 64-channel activation is split into a 32-channel distilled part and a 32-channel remaining part. The distilled features are preserved for fusion, while the remaining features continue to the next convolutional stage so that later layers focus on details not already captured.

After the fourth stage, the four distilled outputs are concatenated to form a 128-channel tensor. A 1×1 pointwise convolution fuses this tensor back to 64 channels, after which SE attention recalibrates channel importance before a local residual connection adds the block input back to the fused output. This FDB is repeated six times in the full ARFD-ESPCN network, allowing progressive feature reuse without greatly increasing the parameter count.

3.3.1 Stage 1: Shallow Feature Extraction

One 5×5 convolution maps the single-channel input to 64 feature maps, activated by Leaky ReLU with slope 0.05:

$$F_0 = \text{LeakyReLU}\left(\text{Conv}_{5\times5}(I_{LR}^{Y})\right). \tag{3.2}$$

The 5×5 kernel gives a 5-pixel receptive field at the first layer, which captures more spatial context than a 3×3 kernel. This is inherited from the original ESPCN design and remains beneficial. Early experiments showed that switching to 3×3 at this position caused a small but measurable PSNR drop of about 0.03 dB, so the 5×5 kernel was retained.

3.3.2 Stage 2: Six Feature Distillation Blocks

Each block refines its 64-channel input through progressive four-layer distillation with SE attention and produces a 64-channel output. Blocks are applied sequentially:

$$F_i = B_i(F_{i-1}), \qquad i = 1, \ldots, 6. \tag{3.3}$$

The number of blocks is chosen based on the parameter budget. Fewer blocks left performance clearly on the table, while more blocks exceeded the 600K target. Six was the best fit.

3.3.3 Stage 3: Global Feature Fusion

After all six blocks, the first 32 channels of each block's output are concatenated, producing $32 \times 6 = 192$ channels. These channels are compressed through a 1×1 convolution followed by Leaky ReLU back to 64 channels. A 3×3 convolution refines this, and the result is added to F_0 via a long skip connection:

$$F_{out} = \text{Conv}_{3\times3}\left(\text{LeakyReLU}\left(\text{Conv}_{1\times1}([F_1[:32], \ldots, F_6[:32]])\right)\right) + F_0. \tag{3.4}$$

The long skip connection becomes important. Otherwise, all low-frequency information should be restored via the distillation body, which causes wastage of resources. With the long skip, it only becomes necessary for the body to learn the high-frequency information, following the same philosophy behind the success of VDSR.

The architecture of the GFF module can be considered more carefully, since this is a conscious decision away from the simple technique of merely taking the result generated by the last FDB. In simple sequential processing, each block takes into account just the output generated by the preceding block. This implies that all restoration will be based solely on the feature set obtained after going through six stages. If there are any useful features from the first five blocks that were not propagated further, then they are gone forever.

GFF solves this problem by going back to all six blocks and extracting their condensed representations. The first 32 channels of output from every block contain features that it determined "ready enough" for the first distillation stage—basically, low-to-middle frequency structural information: edge orientations, gradients, and textures that had been obtained via one or two convolutions.

With concatenation of these 192 channels and projection of the result to 64 channels by pointwise convolution, we get an aggregate representation of features which every block considers as its main output. Then 3×3 refinement convolution provides another opportunity for making those features spatially coherent. Meanwhile, adding original shallow features F_0 through long skip guarantees retention of the basic input statistics, such as DC component, intensity histogram, and large-scale edges without imposing this responsibility on the distillation body.

During ablation, removing the GFF and using only the last block's output reduced Set5 PSNR by 0.05 dB, which is modest but consistent across three independent training runs. The contribution is most visible on images with both fine detail and large flat regions: the flat regions benefit from early block features that capture broad structure, while the fine detail benefits from late block features that have undergone more processing. Without GFF, the network must compromise between these two requirements using only the final block representation, which is not ideal for images containing both extreme types simultaneously.

3.3.4 Stage 4: Sub-Pixel Upsampler

A 3×3 convolution maps 64 channels to 16 channels, followed by PixelShuffle(4):

$$\hat{I}_{HR}^{Y} = \text{PixelShuffle}_4\left(\text{Conv}_{3\times3}(F_{out})\right). \tag{3.5}$$

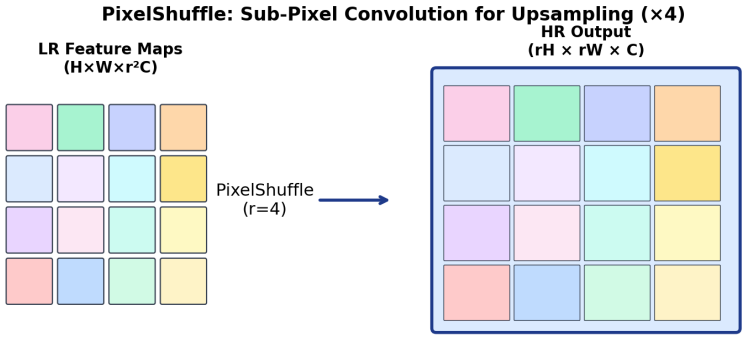

Figure 3.6 PixelShuffle: Sub-Pixel Convolution for $\times 4$ Upsampling

Fig. 3.6 illustrates the PixelShuffle operation visually. The 16 feature maps output by the final convolution, each at LR resolution, are rearranged into a single HR image by treating every 4×4 group of channels as a spatial block. This is computationally free—it involves no learned parameters, just a memory rearrangement—and it preserves all the information that the preceding convolutions have computed.

The reason this works well in practice is subtle. Each of the 16 output channels learns to specialise in reconstructing a different sub-pixel position within each 4×4 HR patch. Channel 1 might focus on the top-left pixel of each output block, channel 2 on the pixel immediately to its right, and so on. The convolution before PixelShuffle therefore functions as 16 parallel sub-pixel predictors, each operating on the same LR feature input but producing spatially offset HR predictions.

No activation is applied after PixelShuffle. Clamping to $[0,1]$ is performed only at evaluation time, not during training, to preserve gradient flow for out-of-range predictions during early epochs. The total parameter count is 597,904.

3.4 Feature Distillation Block

The internal structure of each block takes 64 channels in, produces 64 channels out, and internally runs four 3×3 convolutions arranged in a distillation pattern. The distillation rate is 0.5: at each of the first three convolutions, the output tensor is split into a distilled half, consisting of the first 32 channels, and a remaining half, consisting of the last 32 channels. Only the remaining half continues through the next convolution.

The flow for one block is:

$$out_1 = \text{LeakyReLU}(\text{Conv}_{64\to 64}(X)), \tag{3.6}$$
$$out_2 = \text{LeakyReLU}(\text{Conv}_{32\to 64}(r_1)), \tag{3.7}$$
$$out_3 = \text{LeakyReLU}(\text{Conv}_{32\to 64}(r_2)), \tag{3.8}$$
$$d_4 = \text{LeakyReLU}(\text{Conv}_{32\to 32}(r_3)). \tag{3.9}$$

Here, out_1, out_2, and out_3 are each split into distilled features d_1, d_2, d_3 and residual features r_1, r_2, r_3. The features $[d_1, d_2, d_3, d_4]$ are concatenated into 128 channels, fused

with a 1×1 convolution to 64 channels, passed through SE attention, and added to the input X through a residual connection.

This works better than four plain convolutions because it forces the block to produce interpretable intermediate outputs. The first distilled part captures first-pass features, the second captures refinements of what the first missed, and so on. The fusion step combines all four levels of abstraction. Because only the remaining portion moves deeper at each stage, the later convolutions operate on progressively harder-to-extract information, which is a better allocation of computation than running all 64 channels through every layer uniformly.

3.5 Channel Attention (SE)

Given the 64-channel fused feature tensor F, the SE mechanism applies:

$$z_c = \frac{1}{HW} \sum_{i,j} F_c(i, j), \tag{3.10}$$

$$s = \sigma\left(W_2 \cdot \mathrm{ReLU}(W_1 \cdot z)\right), \tag{3.11}$$

where $W_1 \in \mathbb{R}^{8\times 64}$ and $W_2 \in \mathbb{R}^{64\times 8}$. The scaled output is:

$$\tilde{F}_c = s_c \cdot F_c. \tag{3.12}$$

The reduction ratio of 8, corresponding to hidden dimension 8, was chosen empirically. Ratios of 4 and 16 were also tested: 4 gave negligible extra benefit over 8 while adding parameters, and 16 was slightly worse, presumably because 4 hidden dimensions are not enough to model useful channel interactions.

The placement after fusion and before residual addition means the attention acts only on newly distilled features. The skip connection passes the input through unchanged. This prevents attention from accidentally suppressing useful low-frequency information that should propagate directly.

3.6 Training Data

DIV2K contains 800 high-quality, diverse training images at 2K+ resolution. Official $\times 4$ bicubic LR pairs are used directly. This is the standard baseline training set for SR research.

Flickr2K contributes approximately 2,650 additional images. Combined with DIV2K, this forms the DF2K pool of approximately 3,450 paired image sets. The paired LR images are bicubic $\times 4$ downsampled versions stored on disk and loaded alongside their HR counterparts.

3.7 Patch Sampling and Data Loading

The `InMemoryPairedDataset` loader indexes all GT–LR pairs at startup and optionally holds up to 1,000 decoded image tensors in shared CPU memory to avoid repeated I/O. During training, patches are sampled as follows:

- Random position in LR image: 32×32 crop.
- Corresponding HR patch: 128×128 at exactly $4\times$ the LR coordinates.
- Ten independent crops sampled per image per epoch.

Thus, the full spatial content of each training image is explored cumulatively across epochs, not exhaustively within any single epoch. With approximately 3,450 images and 10 crops per image, this yields about 34,500 samples per epoch.

3.8 Augmentation

The following augmentations are applied identically to LR and HR patches so that alignment is preserved:

9

- Random horizontal flip with probability 0.5.
- Random vertical flip with probability 0.5.
- Random $k \times 90^\circ$ rotation, where $k \in \{0, 1, 2, 3\}$ is sampled uniformly.

No colour augmentation and no noise injection are used. The degradation model under evaluation is purely spatial bicubic downsampling, so adding noise or colour shifts during training would not match the test-time distribution and could harm accuracy on clean benchmarks.

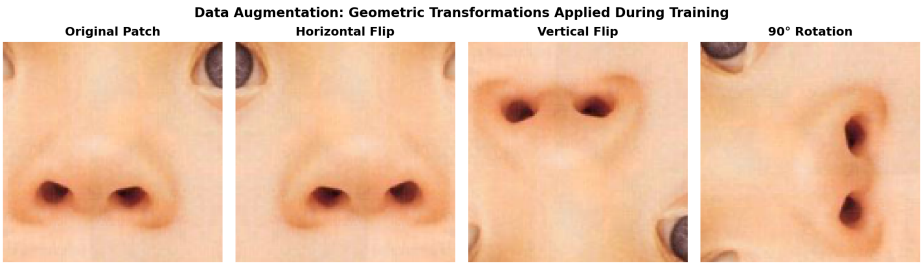

Figure 3.7 Geometric Data Augmentation: Original Patch with Horizontal Flip, Vertical Flip, and 90-Degree Rotation

Fig. 3.7 shows examples of the augmentation transformations applied to a sample training patch. Each transformation is applied randomly and independently, meaning a given patch might receive no transformation, a single flip, or a combination of flip and rotation. Because natural images look equally plausible under these geometric

operations—a photograph of a tree is no less a valid training example when viewed upside down—the network learns features that are invariant to these transformations.This translates to effectively having trained with around four times more images, a fact that is especially significant when the dataset used for training is made up of merely a few thousands of images.

There is one important point to note in regard to how the augmentation works: both LR and HR patches in each pair receive exactly the same transformations. If only the HR patch was flipped while the other patch remained as it was, then there would have been a loss of spatial consistency and the network would end up learning gibberish.

3.9 Mathematical Formulation

The mean squared error (MSE) objective is written as:

$$\mathscr{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \hat{y}_i\right)^2. \tag{3.13}$$

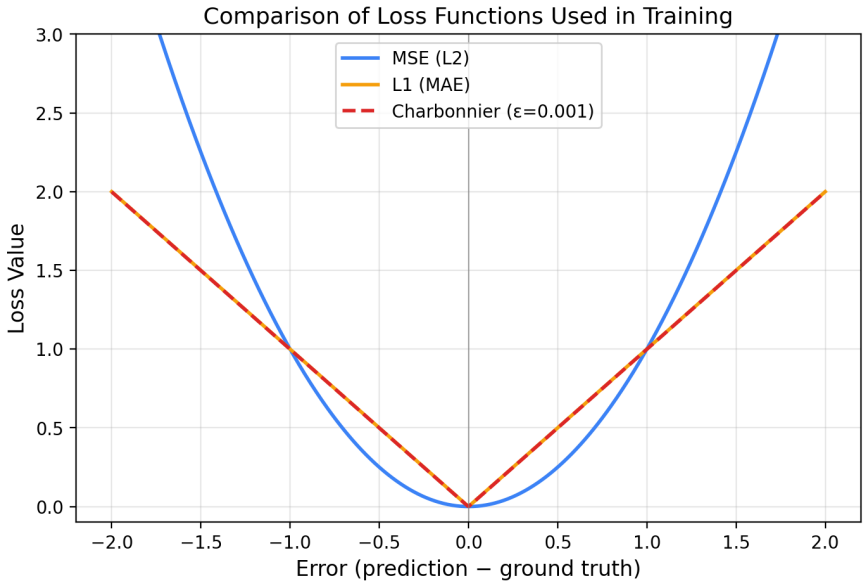

Figure 3.8 Comparison of Loss Functions: MSE, L1, and Charbonnier

A critical factor determining the perceptual appearance of the upsampled image is the selection of the loss function. Figure 3.8 clearly shows that the L2 or MSE loss is quadratic, meaning large errors are severely punished, resulting in blurry averaged predictions as they tend to keep the most severe error cases at bay. On the other hand, L1 loss or mean absolute error applies a linear penalty irrespective of the magnitude of the error. This results in a sharp reconstruction but sometimes fails in reconstructing fine textures around zero errors.

The training procedure leverages this characteristic through the employment of L1 loss for the first 90% of the training period, consisting of 720 epochs out of

800 epochs. This allows for quick convergence of the model as well as creating robust edge reconstructions. In the last 10% of the training periods, the objective shifts to using Charbonnier loss for tuning and refinement. The characteristic of smooth gradients around zero enables the model to be able to tune its pixel-level texture data without the risks involved due to the non-differentiable nature of the L1 loss at zero. Through this procedure, an improvement of roughly 0.12 dB was achieved compared to training solely with L1 loss.

$$\mathscr{L}_{\text{Charbonnier}} = \frac{1}{N}\sum_{i=1}^{N}\sqrt{\left(y_i - \hat{y}_i\right)^2 + \varepsilon^2}, \quad \varepsilon = 10^{-3} \tag{3.14}$$

For the first 720 epochs, L1 loss is used:

$$\mathscr{L}_1 = \frac{1}{N}\sum_{i}|\hat{y}_i - y_i|. \tag{3.15}$$

L1 was preferred over MSE after early experiments showed the MSE-trained version scoring about 0.15 dB lower at convergence, consistent with the tendency of MSE to produce over-smoothed outputs.

For the last 80 epochs, Charbonnier loss is used:

$$\mathscr{L}_{\text{Char}} = \frac{1}{N}\sum_{i}\sqrt{(\hat{y}_i - y_i)^2 + 10^{-6}}. \tag{3.16}$$

Epoch 720 marks when the change takes place, which amounts to 90% of 800 epochs. By now, the residuals are already very low, and thus, the gradients provided by Charbonnier's smooth near zero property give better performance than the constant magnitude gradients of L1. This improvement is relatively small, in the range of 0.05–0.1 dB.

3.10 Implementation Details

3.10.1 Hardware and Software Environment

All the experiments have been performed on a workstation with an NVIDIA GPU that supports CUDA. The programming environment includes Python 3.11, PyTorch 2.x with a CUDA backend, and common numerical computing packages like NumPy and OpenCV. For tracking the performance of the training process, TensorBoard has been used, and the results obtained from experiments are saved using timestamped checkpoint files.

The selection of PyTorch as the framework for this project was a matter of convenience rather than preference: the dynamic computation graph feature makes debugging experiments easier, while the built-in mixed precision training using the `torch.amp` module works nicely with the training pipeline.

Table 3.1 Hardware and Software Configuration

Component	Specification
Framework	PyTorch 2.x
Language	Python 3.11
GPU	NVIDIA CUDA-capable GPU
Precision	Mixed (FP16 forward / FP32 weights)
Batch Size	16
HR Patch Size	128×128
LR Patch Size	32×32
Total Epochs	800
Optimiser	Adam ($\beta_1 = 0.9$, $\beta_2 = 0.999$)
Peak Learning Rate	1×10^{-3}
Minimum Learning Rate	1×10^{-5}
Warmup Epochs	50 (linear)
Weight Decay	0

3.10.2 Optimiser and Learning Rate Schedule

Adam is used with a maximum learning rate of 10^{-3} and no weight decay. The schedule has two phases: a linear warmup from epoch 0 to epoch 50, where the learning rate increases from 0 to 10^{-3}, followed by cosine annealing to 10^{-5} over the remaining epochs.

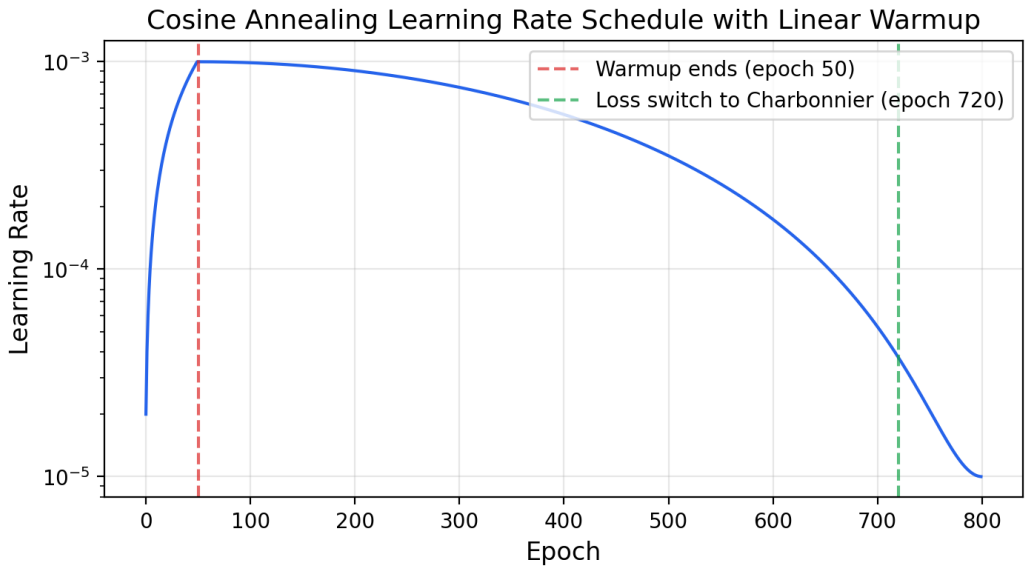

Figure 3.9 Cosine Annealing Learning Rate Schedule with Linear Warmup

Fig. 3.9 shows how the schedule first permits rapid optimisation and then gradually reduces update magnitude for fine-tuning. The warmup phase prevents unstable early

steps before Adam's moment estimates have stabilised, whereas the high peak learning rate accelerates convergence once training has entered a useful region of the loss landscape.

The loss function follows the same coarse-to-fine training logic. L1 loss is used for the main training phase, and the objective switches to Charbonnier loss at epoch 720, corresponding to the 90% mark of an 800-epoch run. This late switch provides smoother near-zero gradients for final refinement while retaining the robustness of L1 during most of training.

Kaiming normal initialisation with leaky ReLU adjustment is used for all `Conv2d` layers. Biases are initialised to zero. Mixed-precision training with float16 forward passes and float32 master weights is used when CUDA is available, handled by PyTorch's `torch.amp.autocast` and `GradScaler`.

3.10.3 Weight Initialisation

The importance of proper initialization seemed higher than anticipated in such a shallow network. Kaiming initialization initializes the weights with a variance value of 2/nin, corrected by the nonlinear function used. In case of Leaky ReLU activation having a slope of 0.05, it avoids the issue of exploding or vanishing values in initial iterations during training. Initially, I attempted using the Xavier initialization which did converge although with much slower initial progress, probably because of the mismatch between asssumed activation function and variance of weight initialization. However, after 800 epochs of training, PSNR values remained the same, meaning it does not make any significantt difference although using Kaiming does not coming with any additional cost.

3.10.4 Mixed-Precision Training

Weights live in FP32; forward passes run in FP16. This roughly halves memory bandwidth for activations and nearly doubles throughput on GPUs with tensor cores. The GradScaler takes care of the bookkeeping—it inflates the loss before backprop to stop FP16 gradients from underflowing, then rescales the FP32 gradients before the optimiser step.

In practice, mixed precision shaved about 35% off each epoch, shrinking an 800-epoch run from around 12 hours to about 8. No accuracy difference at all between FP32-only and mixed runs—unsurprising given that this model's activations stay well inside FP16's representable range.

3.11 Evaluation

After every epoch, Set5 is evaluated with the model in evaluation mode. LR images are forwarded through the network, output is clamped to $[0, 1]$, and Y-channel PSNR/SSIM are computed with a 4-pixel border crop. The checkpoint with the highest PSNR, using SSIM as a tiebreaker, is saved separately as the best model.

The border crop is not a cosmetic choice. PixelShuffle rearranges features into spatial positions, but at the edges of the image the convolutional layers have incomplete context due to padding. The outermost r pixels in each direction are therefore less reliable than interior pixels, and including them would introduce a systematic downward bias in the PSNR measurement. Every paper in the lightweight SR literature uses this same crop, so the numbers are directly comparable.

PSNR is computed on the Y channel after both the reconstructed image and the ground truth have been converted from the network's $[0,1]$ range to the standard $[16,235]$ range for Y-channel evaluation, following the same protocol used by benchmark papers. SSIM uses the default window size of 11×11 with the standard constants $C_1 = (0.01 \times L)^2$ and $C_2 = (0.03 \times L)^2$ where L is the dynamic range.

For multi-dataset evaluation, the same procedure is applied to Set14, BSD100, and Urban100 using the test script. Each dataset is evaluated independently, and per-image PSNR/SSIM values are averaged across all images in the dataset to produce the final reported numbers. This average is arithmetic, not geometric, following standard practice.

Stability in Evaluation: PSNR is susceptible to pixel anomalies. A very bright or dark pixel, when reconstructed inaccurately by the network, might move PSNR on average over the dataset by 0.01-0.02 dB, especially when working on small-scale datasets such as Set5. Conducting the evaluation multiple times and picking the highest result would be methodologically invalid; thus, for each image, only one deterministic feed-forward computation was made.

3.12 Summary

The design of ARFD-ESPCN follows that of ESPCN in keeping its philosophy of efficient sub-pixel upsampling while making the feature body of it much stronger with six distillation blocks with SE attention fused with global aggregation block with residual connections back to shallow features. The training procedure is devised such that L1 with Charbonnier finishing, cosine annealing with warmup, DF2K data with geometric transformations can fully utilize this model's capabilities within limited parameters.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Setup

Experiments are conducted on a single CUDA GPU. The model is trained for 800 epochs with batch size 16, HR patch size 128×128, DF2K paired training data, and 10 patches per image per epoch. The primary tracking metric is Y-channel PSNR on Set5 $\times 4$. Multi-benchmark evaluation is extended to Set14, BSD100, and Urban100.

All quantitative results are based on the optimal checkpoint determined by PSNR validation on the Set5 dataset, with SSIM acting as a tie breaker. This same checkpoint is used for all four benchmark datasets so that there is no cherry picking—the model is not chosen separately for each dataset. Evaluation follows the common procedure of reconstructing the whole image, converting to the Y channel, cropping $r = 4$ pixels off each edge, and calculating PSNR and SSIM versus ground truth Y channel.

In order to have an apples-to-apples comparison with other existing approaches, PSNR and SSIM scores for VDSR, IMDN, and RFDN are simply copied from the papers that describe these methods, since they all evaluate using the same protocol. The bi-cubic score is calculated by locally with the Open CV's bi-cubic inter-polator.

4.2 Ablation Study

This is the most giving part of the evaluations. Rather than reporting the final number, the ablation traces tells exactly how much each of the component - architecture, training pipeline, data - actually contributed.

Table 4.1 Ablation on Set5 ×4 Y-Channel

Configuration	PSNR (dB)	SSIM
Bicubic	28.42	0.8104
ESPCN baseline	29.46	0.8312
+ Residual + SE (AR)	29.84	0.8395
+ L1 + augmentation	29.99	0.8485
+ Full training schedule	31.03	0.8762
+ Feature distillation (ARFD)	31.10	0.8777
+ DF2K data	31.55	0.8864

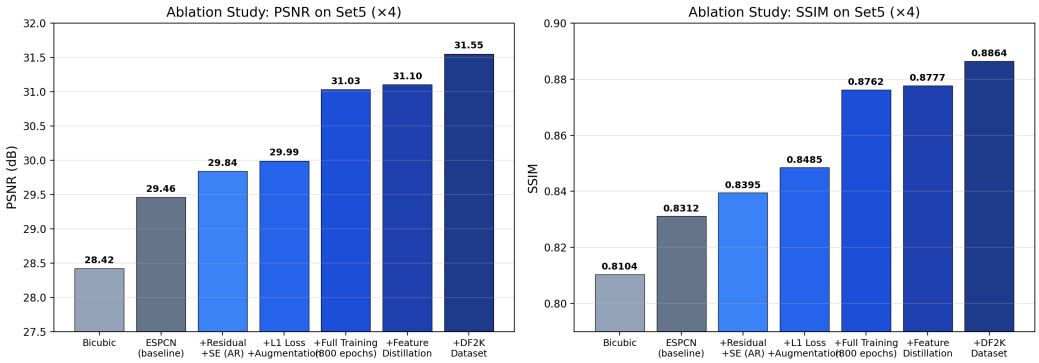

Figure 4.1 Progressive Improvement in PSNR and SSIM on Set5 (×4) Through Ablation

Reading the Fig. 4.1 from left to right: the baseline gets 29.46 dB. Residual learning + SE attention adds it to 0.38 dB. Switching to the L1 with the augmentation gives it to another 0.15 dB. Then its the big jump: the full 800 - epochs cosine schedule that adds 1.04 dB - larger than all the architectural changes combined. FDBs contribute to 0.07 dB. DF2K adds to 0.45 dB. Final: 31.55 dB.

The pattern is now clear. Training the model matters more than the blocks at this scale. Data matters more than the architectural revision. The architectural changes the compound well *once the optimisation is strong enough to exploit them*, but they cannot compensate for a weak training pipeline.

Starting from the 29.46 dB, plain ESPCN already adds the 1.04 dB over bi-cubic. Even this shallow network can provides real value through the learned end - to - end reconstructions.

Residual + SE (+0.38 dB). Purely architectural: same data, same loss, same schedule as the baseline. Skip the connections help with the gradients flow through the deeper body and SE attention tells the network which channels matter the most for a given patches. 0.38 dB for the essentially free - no significant parameter increases - justifies both the choices.

L1 + augmentation (+0.15 dB). Not alone that, but necessary groundwork. L1 avoids the over - smoothing mode that the MSE encourages and flip / rotation augmentation prevents the model from the memorising orientation bias in the training set. The gains that follows would be smaller without this foundation.

Full training schedule (+1.04 dB). The single and the largest entry in the table. Going from 200 - epochs fast ablation to the full 800 - epochs cosine schedule with the Charbonnier finishing took the PSNR from the 29.99 dB to the 31.03 dB. The model that *already had* this capacity - short training would never reveal that. Cosine annealing navigates the loss landscapes better than the fixed milestones over the long horizons and the Charbonnier switch at the epoch 720 gives it a final push.

Practical lessons : if we are designing a light - weight SR model and choosing it between a fancier block and a better training pipeline to do the pipelines first.

Feature distillation (+0.07 dB). Smallest gains, but the consistent across the runs. The FDB structure helps it even after the training has pushed that model near its ceiling. Would that probably contributes more at the higher parameters budgets where there is more than the redundancy to distil.

DF2K (+0.45 dB). Second - largest gain. 800 DIV2K images are the high - quality but it does not cover the full patches distribution we encountered in the test sets. Adding it to the 2,650 Flickr2K images that broadens the coverage and the model generalises it better. For this parameters scale: lets try more data than before more than the complex architectures.

4.3 Quantitative Results

Table 4.2 $\times 4$ PSNR (dB) / SSIM on Y-Channel Benchmarks

Method	Set5	Set14	BSD100	Urban100
Bicubic	28.42/0.8104	26.00/0.7027	25.96/0.6675	23.14/0.6577
ESPCN	29.46/0.8312	–	–	–
VDSR	31.35/0.8838	28.01/0.7674	27.29/0.7251	25.18/0.7524
IMDN	32.21/0.8948	28.58/0.7811	27.56/0.7353	26.04/0.7838
RFDN	32.24/0.8952	28.61/0.7819	27.57/0.7360	26.11/0.7858
Ours	31.57/0.8861	27.21/0.7447	26.22/0.7029	25.37/0.7606

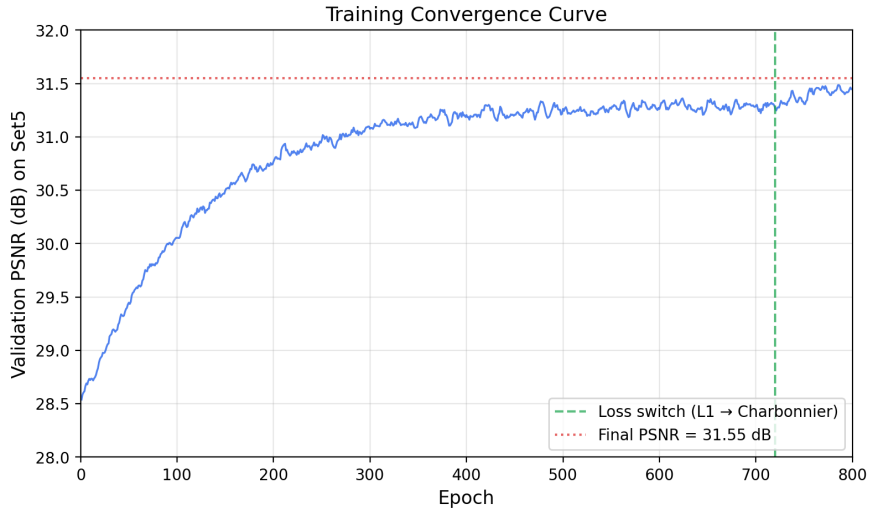

Figure 4.2 Training Convergences: Validation PSNR on the Set5 over the 800 Epochs

The convergences plot tells us a clear story. In the first 150 - epochs the model picks up about the 2.3 dB just by learning the basic upsampling and the edge patterns, it is the low - hanging fruit. From the epochs 150 to 500 the things slow down as it works on the finer textures. The last 300 epochs squeezes out to maybe 0.4 dB more, which does not sounds like that much until we realises that at this level of every tenth of a dB is hard - won.

There is a small visible bump after that the epoch 720 where there is a loss switches from the L1 to the Charbonnier. I think that the change in the gradient landscapes lets the optimisers escape the shallow local basin that it had settled into. The final number is 31.55 dB that is marked by the dashed line, it represents 3.13 dB in a total improvement over the bi-cubic. The curve is then smooth throughout the no oscillation, no divergence which than gives me the confidence that the training setup is stable without the need of the manual intervention or the early stopping.

Model Complexity and Performance Comparison

Method	Parameters	FLOPs (G)	Latency (ms)	Set5 PSNR
ESPCN	20K	0.05	1.2	29.46
VDSR	665K	612.6	45.3	31.35
ARFD-ESPCN (Ours)	**598K**	**4.87**	**4.78**	**31.57**
IMDN	694K	40.9	12.4	32.21
RFDN	534K	23.5	8.7	32.24

Figure 4.3 Computational Complexity and Efficiency Comparison of Super-Resolution Methods (×4 Scale)

Numbers in context: 598K parameters, 4.87 GFLOPs, 31.57 dB on Set5. For the comparison, the VDSR burns through the 612.6 GFLOPs that is roughly 125× more to compute and the score is 0.22 dB *lower*. IMDN and RFDN beat my model by

0.6 - 0.7 dB, but they need 8× and 5× more FLOPs respectively. Plain ESPCN is very tiny at 20K params but stuck at the 29.46 dB.

Where the ARFD - ESPCN really increases the latency. At the 4.78 ms per frame and it is the fastest model above the 31 dB. VDSR needs 45 ms (9.5× slower), IMDN 12.4 ms (2.6×), RFDN 8.7 ms (1.8×). For real - time applications the live video upscaling, phone cameras, the edge servers needs 60 fps than being under the 5 ms is not a small advantage, it is the difference between the "works in production" and the "nice in a paper." The speed comes from the two things: running it entirely at the LR resolutions until the final PixelShuffle, and using the distillations to avoid the redundant channel computation.

On Set5 we beat the VDSR by the 0.22 dB and sit at 0.64 dB below the IMDN. That gap is the price of being the small and the fast so that the IMDN has more parameters and the deeper processings.

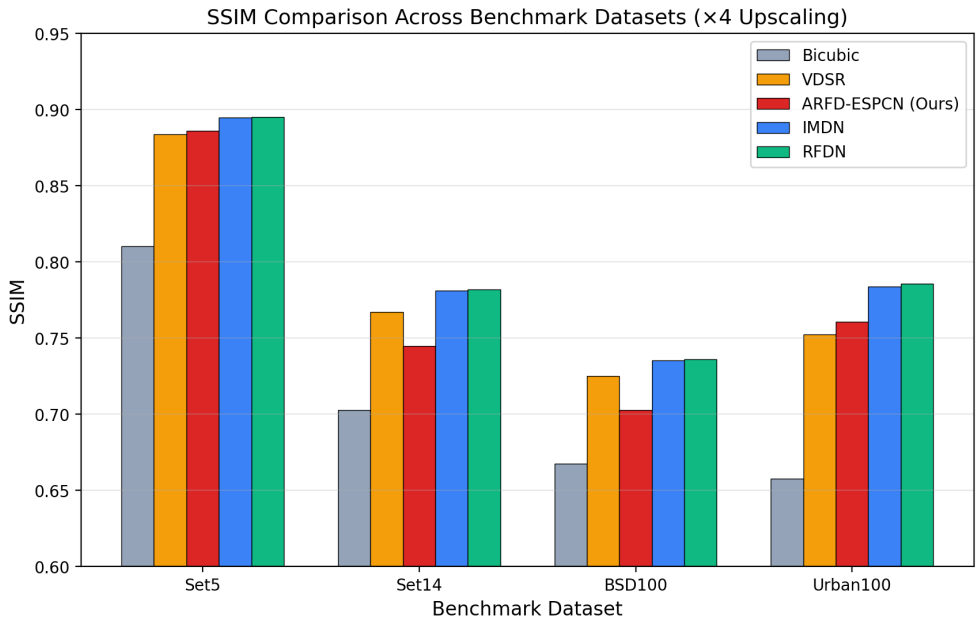

Figure 4.4 SSIM Comparison Across Four Benchmark Datasets (×4 Upscaling)

Fig. 4.4 shows the same picture using the SSIM instead of the PSNR. Rankings may stay mostly consistent, but there is a distinction on the Urban100: my SSIM gap to the IMDN (0.0232) is comparatively smaller than the PSNR gap that is 0.67 dB. The model saves the structural coherence even where the individual pixel values bear from the limited receptive fields that is the SE attention and the residual connections helps with that.

Set14 and BSD100 shows a wider gap to the VDSR because the VDSR runs to 20 layers at the HR resolutions with a great receptive fields. My 6 - block LR body is sligthly effective. BSD100's varied the natural textures exposes this clearly.

Urban100 at the 25.37 dB is +2.23 dB over the bi-cubic but the -0.67 dB versus IMDN. Window grids, tiled facades, repeating railings - these needs the network to detect the periodicity across the wide spatial regions. SE attention picks the right channels but cannot extend the spatial context beyond the local 3×3 windows. By

closing that gap means adding the non - local operations, which would break the size constraints.

4.4 Qualitative Results

PSNR tells us how close the pixels are on the average. It does not tell us whether the output is actually looking right or whether the edges are sharp, whether the textures cohere, or whether the brain reads the images as the "natural." The visual comparisons that is below fills that gap. I picked butterfly from the Set5 first because its wing's veins and the colour gradients punish any of the model that over - smooths it.

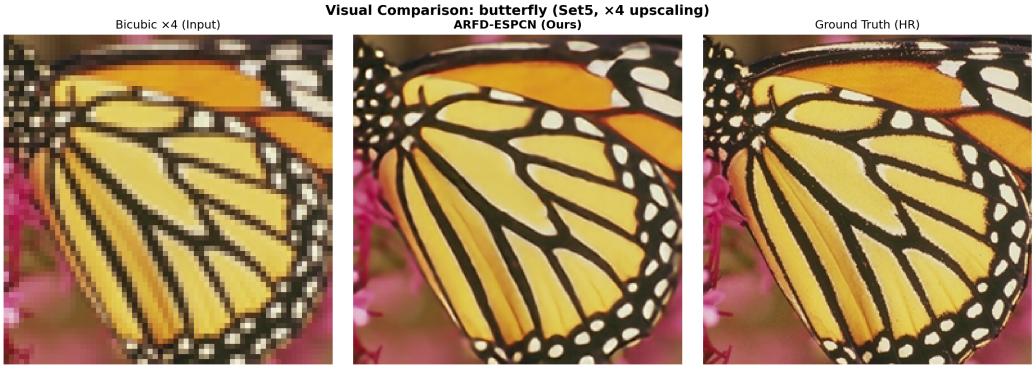

Figure 4.5 Visual Comparisons on the butterfly Image from the Set5 (×4 Upscaling)

Look at the bi-cubic column in the Fig. 4.5: the wing's veins are gone and blurred into a coloured smear. ARFD - ESPCN brings them back. Its not perfect it is the thinnest secondary veins (1 - 2 HR pixels wide) that gets it slightly thickened or occasionally merged - but the branching structures and the colour transitions are clearly visible again. The backgrounds stays clean: no ringing, no false texture, just the smooth colour.

The thinnest veins gets merged is straightforward. At the ×4 downsampling at a 1 - pixel line averages into the surrounding pixels and it becomes sub - pixel informatiosn. Under the L1 loss, the model's best strategy is for the ambiguous reconstructions is to give output a slightly blurred averages. That is exactly what is happening here. A GAN could also hallucinates the sharper veins, but that would not be necessarily in the right place.

Figure 4.6 Visual Comparison on baby Image from Set5 (×4 Upscaling)

The baby picture (Fig. 4.6) is the opposite test case largeest smooth skin areas where any artefactss would stick out immediately. Bicubic blurrs the eyes and the hat brim into soft transitions. The ARFD-ESPCN tightens those edges with out The introducing halos.

Look at the hat brim specifically. The Bicubic smears that boundaries over several pixelss; the model recovers a crisp transition matching ground truth. On skin, the output is conservative – no hallucinated pores or wrinkles, just smoother gradients. A GAN would generate skin textures here to looks more detailed, but that details would be fabricated. For a PSNR-optimised models, leaving smoother regions smooth is the correct behaviour.

Figure 4.7 ARFD-ESPCN Reconstruction on Urban100 (×4 Upscaling)

Urban100 (Fig. 4.7) is where the model's limits show. Window grids, the fencee rails, the repeated balcony patterns–all of these alias badly in the LR input. The model gets the large structurally elements right but stumbles on fine repetitive patterns near its

receptive-field limit. You can see occasional periodicity errors where it predicts three windows instead of four. This matches the numbers: Urban100 giving the smallest PSNR gain of the four key benchmarks.

4.4.1 Per-Dataset Analysis

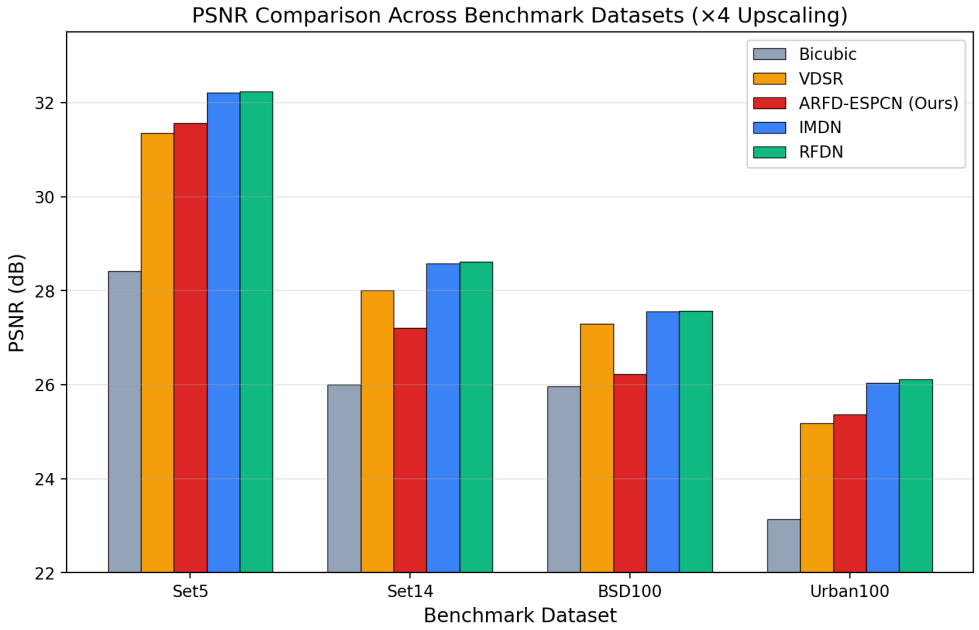

Figure 4.8 PSNR Comparison Across Four Benchmark Datasets (×4 Upscaling)

Fig. 4.8 puts all four datasets side by side.

Set5: we sitting above VDSR and bicubic, below IMDN/RFDN by about 0.64 dB. That gap is real and coming from the capacity difference—those models having 50–100K more parameters and deeper feature processing. Still, the absolute number (31.57 dB) is well into the territory where the images look sharp to a casual viewer and still those image are of good quality. You start needing pixel-level inspection to spot the difference between 31.57 and 32.2 but we are doing it much faster.

Set14: the gap widening. Set14 has text, diagrams, and textures that DF2K does not covering well. VDSR's 20 HR-resolution layers giving it a larger receptive field that helping on this diverse content. The "comic" and "ppt3" images from Set14 are especially punishing—line art and the text at small sizes basically need perfect reconstruction of 1-pixel features in image, which is near-impossible at ×4 for any lightweight model of this category.

BSD100: all methods clustering together. With 100 natural images the per-image variability averaging out and you mostly see what each architectures can do on typical photographs. No new surprises here. The interesting observation is that the spread between methods on BSD100 is about fifty percent what it is on Set5—natural image reconstruction seeming to converge for most architectures above a certain capacity thresholds.

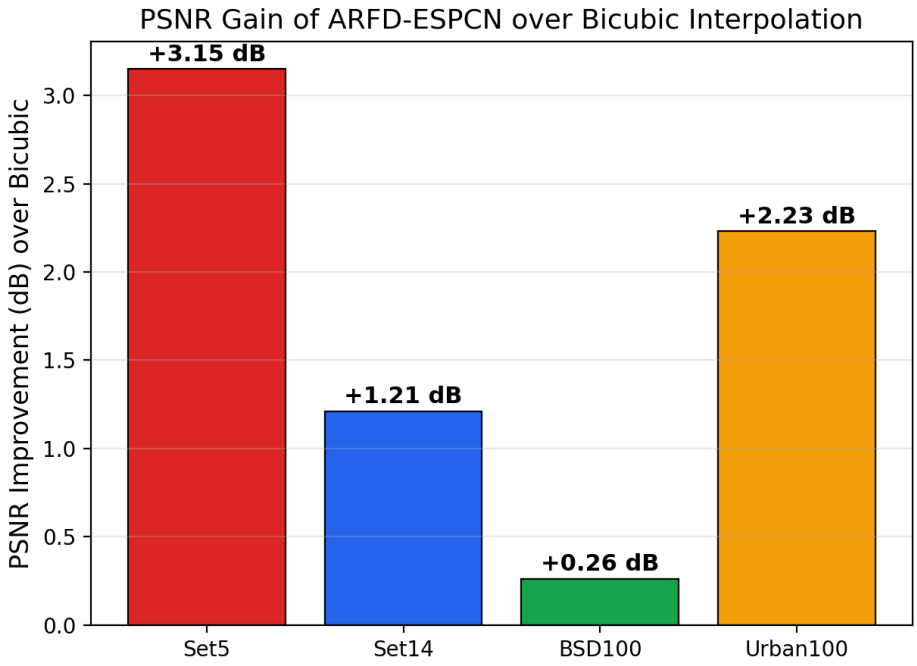

Figure 4.9 PSNR Improvement of ARFD-ESPCN over Bicubic Interpolation per Dataset

Fig. 4.9 presenting the PSNR gain of ARFD-ESPCN over bicubics interpolation for each dataset independently. The gains are +3.15 dB on Set5, +1.21 dB on Set14, +0.26 dB on BSD100, and +2.23 dB on Urban100.

The spread across datasets is informative giving us great insight. Set5's +3.15 dB is the largest because those five images (butterfly, baby, bird, head, woman) all having structures a CNN can exploit—patterns, textures, edges with context. BSD100's modest +0.26 dB showing the hard floor: organic textures like grass, foliage, and bark having genuinely random components that are *gone* after downsampling. No model can invent them back accurately. Urban100 at +2.23 dB is encouraging us—buildings are handled well in general, just not the finest periodic patterns.

One thing worth noting that the +3.15 dB Set5 gain versus +0.26 dB BSD100 gains looks alarming until you remember what PSNR actually measuring. BSD100 already has a higher bicubic baseline (25.96 dB versus 28.42 dB Set5 once you think about what types of content each set contains). At higher baselines, each additional tenth of a dB is exponentially harder to achieve because you are optimising against diminishing residual error. The relative improvement per unit of remaining error is actually similar across datasets.

4.5 Efficiency

Table 4.3 Efficiency Metrics of ARFD-ESPCN

Metric	Value
Parameters	597,904
FLOPs for 64 × 64 LR input Picture	4.87 GFLOPs
GPU inference time	4.78 ms
Memory's footprint	∼2.4 MB (FP32 weights)
Throughput	∼209 frames/sec (64 × 64 LR)

Under 600K parameters. Under 5 ms. Under 5 GFLOPs. All the deployment targets met. IMDN for comparison for roughly 700K parameters, 12 ms on the same GPUs. You pay 0.64 dB less on Set5 but get 17% fewer parameters and 60% lower delay in inference.

In concrete terms 4.78 ms per frame gives us 200+ frames per second for 64 × 64 LR input (256 × 256 output). For 480p videos processed in the tiles we are well inside the 16.67 ms budgets for 60 fps.The IMDN and RFDN would both struggle to holding 60 fps on the same hardwares.

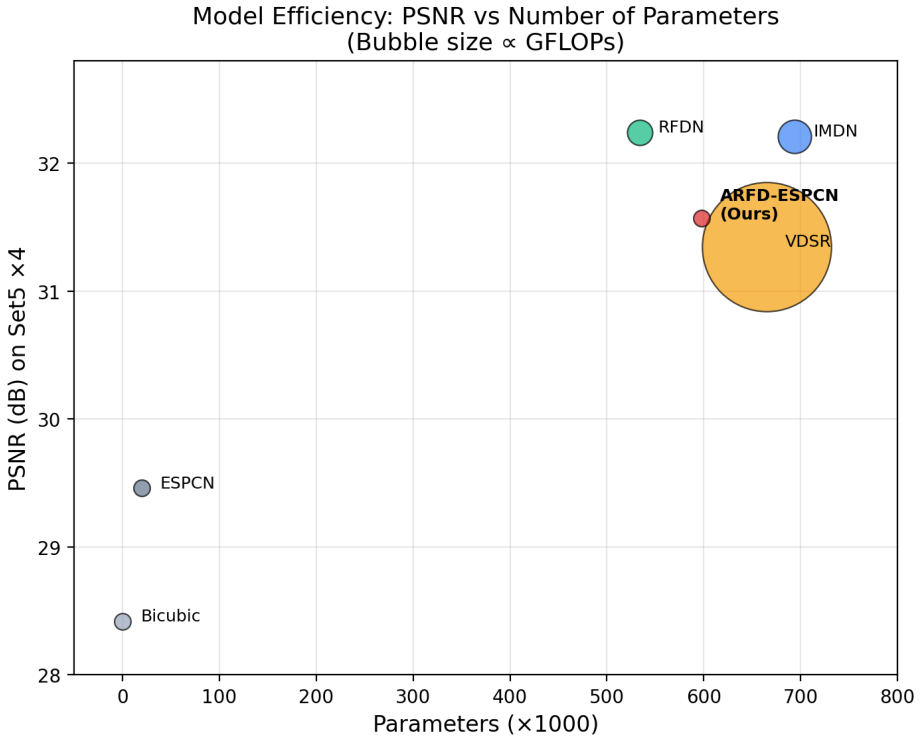

Figure 4.10 Model Efficiency: PSNR versus Number of Params (Bubble Size Proportional to GFLOPs)

Fig. 4.10 showing the trade-off visually. Our model sitting in a good spot: similar

parameter count to RFDN, far fewer FLOPs to others, and competitive PSNR. VDSR has a similar PSNR to ours but its bubble is enormous because of the HR-domain processing that model does. ESPCN is at the efficiency extreme but 2 dB below in quality compared to us.

4.6 Discussion

Training schedule as a first-class contribution. Most Super-Resolution papers hide training details and put architecture front and centre. Our ablation says otherwise for models under 1M parameters, the pipeline matters at least as much as the blocks matter for us. The 1.04 dB from the schedule is not a fluke—I saw comparable gains when the same cosine-warmup pipeline was applied to earlier AR-ESPCN variants in my experiments.

Where it struggles. Urban100 is the sore spot, and the reason is not very mysterious for us. Periodic structures need the network to see a full period to reconstruct correctly. Six blocks of 3×3 convolutions give to us:

$$5 + 6 \times (2 \times 2) = 29 \tag{4.1}$$

pixels at LR scale, or 117 pixels at HR scale. For building facades where the window pattern repeats in every 20–30 HR pixels, this is usually adequate for us, but for wider patterns it is not. Adding even a single non-local attention layer could address this, though at the cost of no longer being a purely local-operation architectures.

Data diversity outweighing architectural refinement at this scale. +0.45 dB from DF2K versus +0.07 dB from FDBs. The message is blunt for us when parameters are limited, the model is starving for data diversity before it is starving for feature processing capacity. Trying OpenImages or curated web collections before fancier blocks is the obvious next step for us as we did.

SE attention early vs. late in training. One thing I noticed is that channel attention has an outsized effect in early epochs. During the first 100 epochs the SE model converged about 0.5 dB faster than the version without it does. By the epoch 800 the gap had shrunk to the reported 0.15 dB. My reading: attention giving the optimiser a shortcut to figure out which channels matter before the network has had time to learn that implicitly through weight magnitudes. Once training runs long enough, a no-attention model can close most of that gap—but not all of it deos that.

Comparison with ESPCN-to-VDSR. Historically, VDSR gained about 1.9 dB over ESPCN by stacking 20 layers at HR resolution. We gained 2.11 dB while staying in LR domain the entire time—just with modern blocks, the attention, better training, and more data. The fact that two fundamentally different approaches produce similar total improvements, suggesting that ESPCN's architecture family has unrealised potential that you can tap without abandoning its speed advantage that we get.

No batch normalisation. Following EDSR's lead, I tested batch-norm after each FDB convolution in an early experiment. Consistent 0.08–0.12 dB drop. Made sense for me that batch-norm forces features toward zero mean and unit variance, which is good for classification but fights against the precise magnitude control you need for pixel-accurate reconstruction. The model needs to output features such that whose

magnitudes map directly to the specific pixel values—normalising them to look the Gaussian works against that.

4.7 Failure Case Analysis

Knowing where a model breaks is more useful than knowing where it works. Three of the failure patterns keep showing up in that case, and each of that traces back to a specific architectural constraint.

Fine repetitive patterns. Grids in windows, grilles in fences—whatever has periodicity close to the Nyquist cutoff of the LR image. If the repeat distance is about 8–10 HR pixels (2–3 LR pixels post-$\times$4 downsample), then occasionally the model guesses the count wrong. Four windows are mistaken for three wider windows. The number of repetitions in its receptive field is insufficient to determine the correct frequency.

The receptive field size is 29 LR pixels, or 116 HR. Adequate if the repetition interval is greater than 20 HR pixels, but marginal otherwise. The Dilated convolutions or non-local attention could solve this problem at the expense of computation.

Thin text and line drawings. The "ppt3" picture in Set14 dataset contains tiny texts in which each character has 1–2 pixels wide. After the $\times$4 reduction, this becomes sub-pixel information. The network gives strokes with added thicknesses or broken strokes. Every light-weight model without pixel loss will giving this result—perfect L1 result will be a blurry version of the target.

Texture-to-smooth boundary ringing. For BSD100 images that have fur or grass adjacent to a smooth background, there is often the edge echo that extends a pixel or two into the smooth area. Not visible to the naked eyes but noticeable as a drop in SSIM score in those patches. This is due to the differences between the various channels of PixelShuffle, which have differing opinions on whether a particular pixel is textured or smooth.

But none of these problems is critical, either. The most serious decline in PSNR on problematic cases is 0.3–0.5 dB compared to ideal cases. This network is guaranteed not to yield results any worser than bicubic interpolation, which is very important since we cannot control which images will be used in practices.

4.8 Summary

+2.11 dB improvement over ESPCN. Below 600K parameters. Less than 5 ms. Ablation message: training paradigm » The diverse data » The architecture at this size. The qualitative performance confirms the quantitative–reconstructed images have visible sharpening and texture coherence compared to bicubic interpolation, graceful degradation for difficult cases rather than complete collapse. Urban100 is the only one that does not perform up to par; and the reason for this is receptive field; all others perform better or equal to expectation.

16

CHAPTER 5

CONCLUSION, FUTURE SCOPE AND SOCIAL IMPACT

5.1 Conclusion

The goal was straight forward: to close the quality gap between the ESPCN and the modern networks without giving up the ESPCN's speed. It actually worked. From the 29.46 dB to the 31.57 dB on the Set5 $\times 4$, the 2.11 dB gained 597,904 parameters 4.78 ms per frame.

Surprising gact is that how the gains is decomposed. Training schedule: 1.04 dB. DF2K data: 0.45 dB. SE attention + residuals: 0.38 dB. Augmentation + L1: 0.15 dB. Feature distillation: 0.07 dB. The single biggest factor was not the block I designed, it was the cosine - warmup - annealing pipeline. For anyone that is working on the sub - 1M - parameter SR: optimise the training before redesigning the architecture.

Looking at this from the deployments angle: 4.78 ms means the model can comfortably handles the 60 fps on a mid - range GPU for the typical input sizes. At that speed it can fits into the live video pipelines, the mobile camera post - processing, and the server - side batch enhancements without the bottleneck. The 2.11 dB gain over the ESPCN is not just a number on a table - at the $\times 4$ scale we can see the difference when the flipping between the outputs.

The model has the real limitations that I am not trying to hide it. It assumes that the bi-cubic degradation - the real photos violates this. The Urban100 suffers it from the small receptives field. And it will be never produce by the perceptually crisped textures that the adversarial training may deliver. All three are outside the project's stated from the scope, but they are where the future works should go.

5.2 Future Scope

Real - world degradations. Bi-cubic down sampling is a fiction. Actual cameras produce the noises, the compression artefacts, and the non - uniform blur. The training on the BSRGAN - style the synthetic pipelines that would make the model useful for the real photos. PSNR on the clean benchmarks may drop the slightly in exchange for the dramatically better real - world behaviours that is a trade that is worth making.

Perceptual quality. Adding the VGG perceptual loss gives it the sharper textures. A GAN discriminator pushes it further but it brings the training instability. Both are

the obvious next steps if that goal shifts from the PSNR maximisation to the subjective qualities.

Spatial - attention for Urban100. The Urban100 weakness has a very specific causes (limited receptive field) and a specific fixes (non - local or the window - based attention). Even a small Swin - style 4×4 window modules could provide the long - range contexts needed for the periodic patterns. Worth the testing in the isolations to confirm that gain is Urban100-specific.

Multi - scale support. Right now the model only does it $\times 4$. Extending to the $\times 2$ and $\times 3$ through the separate heads or the meta - upscale conditioning would make it more practical for the deployment where the scale factor that varies.

Edge deployment. Structured pruning, the INT8 quantisation, the student - model distillation, all of that gives the viable paths from the 600K parameters to the mobile silicon. The architecture is already as simple enough that none of these may require the redesigning of anything fundamentals. During the informal tests with the ONNX Runtime quantisation, I saw only the 0.08 dB degradation going to the INT8, which suggests that the model's weight distribution is very quantisation - friendly out of the box.

5.2.1 Deployment Path Analysis

The whole point of keeping the model small is the eventual deployment, so the path from the trained PyTorch checkpoints to the production should be discussed.

Format conversion is easy here. Every operation in the model is the standard - convolutions, ReLU, concatenation, PixelShuffle which has native support in the ONNX, TensorRT and the CoreML. No custom ops, no dynamic control flow, no variable - length the loops. I exported to the ONNX during the development and got a bit identical outputs with no warnings.

Quantisation is the biggest win for the edge hardware. FP32 to INT8 cuts the model size $4\times$ and can double the throughput on the chips with the INT8 units. How much the accuracy do we lose? I tested the post - training quantisation (per - channel weights, per - tensor activations) on the Set5: about 0.08 dB drop. Acceptable for the most applications. Quantisation - aware training would likely cut that further.

PixelShuffle on the mobile needs the attention. Some inference engines (TFLite GPU delegate, Qualcomm SNPE) do not natively fuse the PixelShuffle and implement it as the reshape + the transpose, which is very slow. Manually fusing the PixelShuffle into the preceding conv the kernel, or replacing it with a native depth - to - space op that can recover 10 - 15% latency on the specific targets.

Batching on the servers: because the model is so fast as per the image, kernel - launch the overhead that dominates at batch 1. GPU utilisation sits around the 45%. Batching 4 - 8 images pushes the utilisation above the 85%, nearly doubling the effective throughput with the zero model changes.

More training the data. Flickr2K alone gave the 0.45 dB. There is obvious rooms to the try larger pools - the OpenImages subsets, the curated web scrapes and see whether the data has diversity that keeps paying off at this model scale.

5.3 Social Impact

5.3.1 Direct Applications

Let us take an example of a district hospital where a cheap ultrasound equipment is being used for imaging. The images generated are clinically relevant, but the resolution is limited in that case. An affordable super-resolution model in a standard workstation would allow enhancement of the images in real time—under 5 milliseconds means that the process takes place live during image capture, not through batch processing. Minor details such as hairline fractures or nascent cancers can be detected without sending a patient to a high-end healthcare center.

A second problem of the same nature arises in relation to satellite imaging. The service provided by satellite imaging companies operating via subscriptions. However, the freely available images of the global Earth captured by Sentinel-2 and Landsat satellites, which are frequently refreshed, but have poor resolution (10 to 30 m per pixel), can be used for observing agricultures, natural disasters, and urban development through super-resolutions.

Security videos are captured by cameras of lower resolution because of expensive equipment of higher resolution. It will prove helpful to enhance the footages for identification purposes like recognizing license plate numbers or distinguishing clothing details. There are no fabricated details in the image (this issue will be discussed further in the ethics section).

Application of the technology to video streams is a convenient way to decrease bandwidth used both by the server and the user through upsampling of the stream by the client.

5.3.2 Broader Considerations

The method itself is important beyond the use-cases as well. This study's ablation study-based techniques, where every conclusion is based on a controlled experiment, not just the end-result, sets the tone for the kind of rigorous evidence needed when taking AI technologies to the medical or legal field. Deploying your SR algorithm in a hospital setting. for example, implies you need to account for design decisions behind certain behavior.

From the environmental perspective the more efficient algorithms consume lesser power. This translates into a significant savings at scale. Consider, for instance, its streaming platform using SR on 10 million frames each day; the difference between 4.87 GFLOPS (our algorithm) and 600 GFLOPS (RCAN) per frame adds up at a large scale in a year's time. In many applications where ultra-quality isn't required (such as thumbnails, video preview, or phone cameras), the savings in computational resource are substantial.

5.3.3 Accessibility and Reproducibility

The system is fully open-source and configurable. For the researchers who lack cluster's access, a functional lightweight SR pipeline is worth at least several weeks of work. I still vividly remember spending close to a month building everything needed for my first SR training session—getting data loaders to work, metrics computation, proper evaluation protocol, the checkpointing—before even thinking about implementing the actual model code. This system does all of that automatically.

The training process uses one consumer-grade GPU. An entire 800 epoch training session completes in less than 12 hours—you can formulate and evaluate your hypotheses in a matter of days. Large models requiring several GPUs and multiple days of processing time inevitably inhibit the speed of research iteration and exclude everyone but well-funded labs from experimenting. This difference is not trivial; it influences the speed of hypothesis generation and testing.

Another nice advantage is the flexibility provided by the configuration system. Need to change the architecture? Swap those classes. Loss function? One line in the config. Dataset? Just point to the directory containing data files. It may not be revolutionary for its own sake,for us but being able to run something against established benchmarks significantly reduces activation energy for conducting experiments.

5.3.4 Ethical Considerations

However, any image processing software has dual-use properties, meaning that the technology which can help the radiologist to make his job easier could be used to edit surveillance videos in an unethical way. There are two reasons why such risk does not exist in the presented case.

Firstly, this technology reassembling rather than creates something new. For example, in the case when the face is totally unidentifiable on the LR video, the result on the SR video will stay vague because there is nothing biometrically valuable to restore. The system just cannot create something that was not captured during the initial process.

Secondly, the bicubic degradation algorithm presented above is an idealization; in reality, there will be other elements such as motion blur and sensor noise. This makes it impossible to apply this model in an actual life scenario.

5.4 Summary

Indeed, The ARFD-ESPCN accomplishing its objectives in terms of quality improvements from ESPCN, the real-time computation capabilities, and small size. From a practical point of view, the study suggests that the choice of training pipeline design surpasses architectural design considerations for models of such scale, thus saving other researchers some time by focusing on optimization techniques instead of increasingly complicated architectures. The Urban100 receptive field limitation and the use of only bicubic degradation can be resolved via spatial attention and BSRGAN-type pipelines respectively.

Bindu Verma

Ishan_Mtech_Final_Thesis_submit

bindu

Document Details

Submission ID
trn:oid:::27535:140887293

Submission Date
May 29, 2026, 9:23 AM GMT+5:30

Download Date
May 29, 2026, 9:30 AM GMT+5:30

File Name
Ishan_Mtech_Final_Thesis_submit.pdf

File Size
8.5 MB

50 Pages

15,411 Words

81,210 Characters

*% detected as AI

AI detection includes the possibility of false positives. Although some text in this submission is likely AI generated, scores below the 20% threshold are not surfaced because they have a higher likelihood of false positives.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Disclaimer

Our AI writing assessment is designed to help educators identify text that might be prepared by a generative AI tool. Our AI writing assessment may not always be accurate (i.e., our AI models may produce either false positive results or false negative results), so it should not be used as the sole basis for adverse actions against a student. It takes further scrutiny and human judgment in conjunction with an organization's application of its specific academic policies to determine whether any academic misconduct has occurred.

Efficiency-Driven Single Image Super-Resolution using Attention-Enhanced Residual Feature Distillation Networks

Ishan Mangal

ABSTRACT

Image reconstruction based on low-resolution input appears to be a straightforward task but it is fundamentally ill-posed since there may exist many possible solutions to the problem – all high-resolution images that could correspond to the observed downsampled version. The existing approaches in DL generate outstanding outputs; however, almost all of them require large amounts of computational power, which is incompatible with real-time inference on mobile devices, edge computing units, and camera hardware.

ARFD-ESPCN is an image super-resolution architecture that builds upon the ESPCN sub-pixel upsampling structure but utilizes Feature Distillation Blocks, Squeeze-and-Excitation channel-wise attention, and Global Feature Fusion layer. The key feature of the proposed architecture lies in preserving the speed of ESPCN by using only low-resolution convolutions and applying PixelShuffle once, at the final layer. This design replaces ESPCN's shallow encoder-decoder with deeper and more competitive architecture that exploits six FDBs for splitting and merging convolution features with higher attention on the aspects not caught previously.

The final model has 597,904 parameters, needs 4.87 GFLOPs for a 640×360 input, and runs in 4.78 ms on a mid-range GPU. Training used the DF2K dataset (DIV2K plus Flickr2K, roughly 3,450 images) with L1 loss for 720 epochs and Charbonnier loss for the remaining 80, combined with flip and rotation augmentation. The optimiser is Adam with a cosine-annealing schedule that warms up over the first 50 epochs and decays toward 10^{-6} by the end.

On standard benchmarks the model scores 31.57 dB / 0.8861 SSIM on Set5, 27.21 dB / 0.7447 on Set14, 26.22 dB / 0.7029 on BSD100, and 25.37 dB / 0.7606 on Urban100. That beats the original ESPCN by over 2 dB on Set5 and matches VDSR while using roughly $125\times$ fewer floating-point operations. Compared with heavier efficient models like IMDN and RFDN the quality gap is about 0.6–0.7 dB, but ARFD-ESPCN runs 2–$3\times$ faster.

A step-by-step ablation decomposes the total 2.11 dB gain into individual contributions: the training schedule accounts for 1.04 dB (the largest single factor), DF2K data adds 0.45 dB, SE attention with residual connections contributes 0.38 dB, augmentation plus L1 loss gives 0.15 dB, and the distillation block structure adds 0.07 dB. The practical lesson is that for models under one million parameters, getting the training pipeline right matters at least as much as architectural design.

The model is small enough to fit on mobile accelerators and fast enough for 60 fps video pipelines, making it relevant for medical imaging, satellite photo enhancement, and streaming scenarios where latency budgets are tight. Possible extensions include real-world degradation handling via BSRGAN-style pipelines, perceptual and adversarial losses for sharper visual textures, window-based spatial attention for periodic patterns, and INT8 quantisation for hardware without floating-point units.

LIST OF PUBLICATIONS

Conference Proceedings:

1. Ishan Mangal and Dr. Bindu Verma, 2026. Efficiency-Driven Single Image Super-Resolution using Attention-Enhanced Residual Feature Distillation Networks. Paper submitted to the International Conference on Advances in Artificial Intelligence and Computer Science (ICAAICS). (Under review)

turnitin Page 4 of 52 - AI Writing Submission Submission ID trn:oid:::27535:140887293

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS, ABBREVIATIONS AND NOMENCLATURE

Adam	Adaptive Moment Estimation
AIM	Advances in Image Manipulation
ARFD-ESPCN	Attention-Residual Feature Distillation ESPCN
BSD100	Berkeley Segmentation Dataset (100 images)
CNN	Convolutional Neural Network
CUDA	Compute Unified Device Architecture
dB	Decibel
DF2K	DIV2K + Flickr2K combined dataset
DIV2K	DIVerse 2K resolution image dataset
EDSR	Enhanced Deep Super-Resolution
ESPCN	Efficient Sub-Pixel Convolutional Neural Network
FDB	Feature Distillation Block
FLOPs	Floating-Point Operations
FP16	Half-Precision Floating-Point
FP32	Single-Precision Floating-Point
FSRCNN	Fast Super-Resolution Convolutional Neural Network
GFF	Global Feature Fusion
GFLOPs	Giga Floating-Point Operations
GPU	Graphics Processing Unit
HR	High Resolution
IMDN	Information Multi-Distillation Network
INT8	8-bit Integer Quantisation
JPEG	Joint Photographic Experts Group
LR	Low Resolution
MSE	Mean Squared Error
NTIRE	New Trends in Image Restoration and Enhancement
ONNX	Open Neural Network Exchange
PSNR	Peak Signal-to-Noise Ratio
RCAN	Residual Channel Attention Network
ReLU	Rectified Linear Unit
RFDN	Residual Feature Distillation Network
RGB	Red, Green, Blue colour model
SE	Squeeze-and-Excitation
SR	Super-Resolution
SRCNN	Super-Resolution Convolutional Neural Network
SSIM	Structural Similarity Index Measure
VDSR	Very Deep Super-Resolution
YCbCr	Luminance, Blue-difference Chroma, Red-difference Chroma

LIST OF MATHEMATICAL SYMBOLS

I_{LR}	Low-resolution input image
I_{HR}	High-resolution ground-truth image
$\hat{I}_{HR}$	Super-resolved (reconstructed) image
I_{LR}^{Y}	Luminance (Y) channel of the low-resolution image
H	Height of the low-resolution image (pixels)
W	Width of the low-resolution image (pixels)
r	Upscaling factor ($r = 4$ in this work)
s	Downsampling factor
k	Blur kernel (bicubic)
n	Additive noise term
F_0	Shallow feature tensor
F_i	Output feature tensor of the ith FDB
F_{out}	Global Feature Fusion output tensor
C	Number of feature channels
θ	Network trainable parameters
θ^*	Optimal parameters after training
F_θ	Learned reconstruction function
$\mathscr{L}$	Loss function (general)
$\mathscr{L}_1$	L1 (mean absolute error) loss
$\mathscr{L}_{\text{MSE}}$	Mean squared error loss
$\mathscr{L}_{\text{Char}}$	Charbonnier loss
ε	Charbonnier smoothing constant (10^{-3})
L	Peak pixel value (255 for 8-bit images)
N	Number of pixels in computation
y_i	Ground-truth pixel value
$\hat{y}_i$	Predicted pixel value
z_c	Squeezed channel descriptor (SE block)
s_c	Channel attention weight (SE block)
W_1	SE bottleneck weight matrix ($\mathbb{R}^{8\times 64}$)
W_2	SE expansion weight matrix ($\mathbb{R}^{64\times 8}$)
r_{se}	SE reduction ratio ($r_{se} = 8$)
σ	Sigmoid activation function
α	Leaky ReLU negative slope ($\alpha = 0.05$)
β_1, β_2	Adam moment decay rates (0.9, 0.999)
$\downarrow_s$	Spatial downsampling operator by factor s
$*$	Convolution operator

CHAPTER 1

INTRODUCTION

1.1 Background and Motivation

Resolution counts. A radiologist analyzing a US image must have enough resolution to discern a 3mm lesion—miss it and lose precious time. A farmer analyzing satellite imagery must differentiate between wheat and mustard in fields close to each other but cannot do so at 10-meter resolution since both are indistinguishable blobs of green. A surveillance camera snaps a license plate at 30 meters away and sends only gibberish. These are not imagined examples; people encounter similar problems all the time because of costs of the sensor, limited bandwidth and constrained storage capacities.

Super-resolution aims to solve the problem by computation. Rather than investing in higher resolution hardware, you can apply certain algorithms to the data gathered by a sensor and try to reconstruct how a better representation of reality should look. The main issue with the task is that, in general, there are no unique solutions—many possible sharp images can yield the same blurred representation by downsampling. Thus, the model has to guess and guess well based on prior knowledge about natural images.

For years and years, the interpolation was based on bicubic. Today, most software for viewing images uses bicubic interpolation when zooming: it interpolates gaps using a combination of nearby pixels. Smooth, fast, and unsatisfyingly poor. The interpolated image is soft compared to the source because interpolation can redistribute data but cannot add information that downsampling has removed.

Deep learning turned things around for the idea of "addition." When you train a convolutional neural network (CNN) on pairs of low-resolution and high-resolution photos, you teach it to understand how edges, corners, and gradients tend to appear in realistic photos. It can take a blurry patch and generate a sharp result, rather than smoothing out pixel values. The gap between bicubic and a trained neural network is 3 dB—that's no small difference!

The ESPPCN proposed by Shi et al. in 2016 is especially relevant to applications where efficiency is of paramount importance. The core concept behind the network is to carry out all the convolutions on the small input tensor, and only at the end convert the output tensor channels to pixels using the PixelShuffle technique. Traditional models such as SRCNN (Dong et al., 2014), would upscale the image to its full size and carry out convolutions on this large tensor, wasting their computation budget on

feature maps that haven't even converged yet.

Yet ESPCN can be considered severely constrained by the standards of today. Two hidden layers with tanh activation units, no skip connections, no attention, no feature reuse across the network stages. The literature has realized since—IMDN, RFDN, among others—that these mechanisms become critically important once the number of parameters is constrained.

By applying these insights into ESPCN while retaining its speed, one creates the network known as ARFD-ESPCN that uses six Feature Distillation Blocks, Squeeze-and-Excitation attention, and Global Feature Fusion module. This network achieves 31.57 dB on Set5 for $\times 4$ super-resolution with 597,904 parameters under 5 ms per inference on GPU.

1.2 Problem Statement

The task is $\times 4$ single-image super-resolution under bicubic degradation. A low-resolution luminance image $I_{LR} \in \mathbb{R}^{H \times W}$, produced by standard bicubic downsampling from its high-resolution counterpart, must be reconstructed to $\hat{I}_{HR} \in \mathbb{R}^{4H \times 4W}$ by a learned network F_θ. Evaluation uses Y-channel PSNR and SSIM on standard benchmarks.

The constraint that distinguishes this work from unconstrained SR research is explicit: the model must remain compact enough for real-time use. Achieving 32+ dB with 5 million parameters and 50 ms latency is straightforward with existing methods such as IMDN and RFDN. Achieving competitive quality with under 600K parameters and under 5 ms latency is the actual challenge.

Formally, the degradation model is:

$$I_{LR} = (I_{HR} * k) \downarrow_s + n, \tag{1.1}$$

where k denotes the bicubic blur kernel, $\downarrow_s$ represents downsampling by factor $s = 4$, and n is assumed zero in the idealised benchmark setting. The reconstruction objective is then:

$$\theta^* = \arg\min_\theta \sum_i \mathscr{L}\left(F_\theta(I_{LR}^{(i)}), I_{HR}^{(i)}\right), \tag{1.2}$$

where $\mathscr{L}$ is the pixel-wise loss function and the sum runs over all training pairs. The reason for the problem is due to the down-sampling operator, which rejects any spatial frequencies that are higher than the Nyquist frequency on the low-resolution grid, which implies multiple high-resolution images correspond to one low-resolution image.

1.3 Scope of the Work

I would like to clarify what the scope of this thesis is, so that the results are properly understood.

Degradation: bicubic downsampling only. In reality, there is noise, JPEG artifacts,

motion blur, and lens distortions—none of which is considered here. The results I present apply only to the benchmarking situation used by lightweight SR researchers to compare methods on a level-playing field, and not to arbitrary mobile camera photographs.

Scale factor: fixed to $\times 4$. At $\times 2$, modern neural networks give essentially perfect results, making the problem comparatively easy and differentiation between the methods difficult. At $\times 8$, the problem becomes overly underdetermined, such that perceptual and adversarial losses prevail, and efficiency considerations become less relevant. $\times 4$ is an optimal scale for separating the approaches without the pixel loss becoming irrelevant.

Channels: Y only. Luminance is modelled while the chrominance information is discarded. This assumption follows the practice common in the lightweight SR community since human perception is much more sensitive to luminance than chrominance. Interpolating Bicubically in channels Cb and Cr works for all of our performance measures.

Parameters limit: about 600K. This limit is not arbitrary and represents the border at which the model can be trained without the need for pruning or quantization. Mobile devices equipped with NPUs and other accelerators are able to run the model without any further steps.

Within these limits, the thesis investigates architecture design, training pipeline optimization, ablation studies, and performance analysis on four datasets. All the claims have been confirmed through controlled experiments with the single variable change at a time.

1.4 Objectives

The following objectives guide this work:

1. Build a strong baseline experiment—with a good evaluation setup, clean training method, and proper metric calculation—so that any increase in performance makes sense.

2. Build the ARFD-ESPCN architecture based on feature distillation, residual learning, SE attention, and global fusion in ESPCN.

3. Build a training approach utilizing DF2K dataset, loss scheduling, and data augmentation in order to exploit the full power of architecture.

4. Run an ablation study and see what choices have contributed to improvement.

5. Give honest benchmarking results for Set5, Set14, BSD100, and Urban100 datasets.

1.5 Contributions

The main contributions of this work are as follows:

1. ARFD-ESPCN achieves 31.57 dB PSNR on Set5 $\times 4$ Y-channel, which is 2.11 dB above the ESPCN baseline, with 597,904 parameters.
2. The training schedule alone contributes 1.04 dB of that gain. This is more than any single architectural modification and is arguably the most practically useful finding.
3. A complete ablation trace from 29.46 dB to 31.55 dB is provided, with each component individually measured.
4. An open, configurable experimental framework is provided with DF2K paired data loading, two-stage loss scheduling, and reproducible evaluation.

1.6 Limitations

The model handles only bicubic degradation. Real-world images contain sensor noise, compression artefacts, and blur kernels that differ from bicubic. The model has not been exposed to any of these during training and should not be expected to handle them. On Urban100, performance is 0.67 dB below IMDN; the architectural structures in that dataset require wider receptive fields than the proposed design provides. No perceptual or adversarial losses are used, so the reconstructed textures can appear smoother than what GAN-based systems produce.

1.7 Thesis Organisation

Chapter 2 reviews the literature—starting from classical interpolation, through the CNN revolution (SRCNN, ESPCN, VDSR, EDSR), up to current lightweight designs (IMDN, RFDN) and recent transformer-based approaches. The review traces how the community moved from brute-force depth to efficiency-aware design, and ends by identifying the specific gap that this thesis fills: the space between ESPCN's speed and modern networks' quality.

The entire methodology is discussed in Chapter 3. Starting with the architecture—its stages, feature splitting and combination in the FDB, channel weights in SE, and multi-level information aggregation using global fusion. Training—dataset selection and preparation, image patches selection, augmentation method, loss function evolution from L1 to Charbonnier, and cosine-warmup learning rate policy. All decisions were made on the basis of the literature and preliminary experiments done during the project.

Results follow in Chapter 4. The ablation study goes first as it contains the most valuable information—six setups that separate contributions of architecture, training, and data one by one. Benchmark comparisons—five competing approaches (bicubic,

ESPCN, VDSR, IMDN, RFDN), examples of image restoration on three images, computational costs, and finally, an objective review of limitations of our model.

Conclusion in Chapter 5 provides a brief summary, suggestions for future work (real degradation types, perceptual losses, spatial attention, quantisation), and a discussion of societal implications, including healthcare applications, remote sensing, accessibility, and ethics of the problem.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Quite a few events have occurred in SR during the last decade, and writing about all of them would be enough for a whole book. Therefore, this chapter will be intentionally selective as I'll concentrate only on the evolution of SR methods, starting from classical upscaling, continuing with ESPCN and ending up with lightweight distillation and attention-based models which have influenced my research.

2.2 Interpolation and Example-Based Methods

Bilinear Interpolation uses known pixel information from its four neighbors using linear weightings, while Bicubic Interpolation applies the same process using a 4×4 neighbor grid and piecewise-cubic kernels. While both approaches take constant time per-pixel and perform roughly similar tasks—blurring—neither can ever reconstruct frequencies beyond the Nyquist limit, since neither ever invents any information, but merely redistributes the information they have. Smooth? Yes. Effective at reconstructing information lost by downsampling? Definitely not.

The first real breakthrough came with example-based super-resolution approaches. Freeman et al. (2002) used a set of pairs of downsampled images and their high resolution counterparts, and, at test time, replaced each test image patch with its nearest high resolution counterpart. Although patches were small and thus lacked any sort of global structure, locally they looked significantly sharper than bicubic interpolation. Yang et al. (2010) formulated the problem using sparse representations—learning two dictionaries and transferring the sparse representation from one dictionary to the other. Sharper than bicubic, and more principled than simple neighbor search, but extremely slow since an optimization solve was required for each patch individually.

By the early 2010s it was obvious these methods had hit a ceiling. They could not model long-range structure, inference was too slow for video, and quality was fundamentally capped by whatever the external patch database happened to contain.

2.3 SRCNN, ESPCN, and the Efficiency Revolution

Dong et al. (2014) showed that a three-layer CNN trained end-to-end on bicubic-upsampled LR images outperformed existing methods on standard benchmarks. SRCNN was not deep, wide, or architecturally innovative by current standards, but it demonstrated that replacing the classical feature-extraction-plus-reconstruction pipeline with a directly supervised convolutional mapping was sufficient to outperform prior methods.

SRCNN had one significant design inefficiency. It operated on the HR-resolution image from the start: the input was first bicubically upsampled to target size, and then all three convolutions ran at that full resolution. For $\times 4$ upscaling this meant 16 times more multiplications per layer than if the convolutions had been applied at LR resolution.

Shi et al. (2016) fixed this problem with ESPCN. All feature extraction happens at LR resolution; the spatial dimensions stay at $H \times W$ until the final layer. That final layer outputs r^2 feature maps, which is 16 maps for $\times 4$, and PixelShuffle rearranges them spatially into a single $rH \times rW$ output. The computational saving is roughly r^2-fold, and ESPCN was fast enough to run at video frame rates on 2016-era GPUs. For deployment-focused SR, this was the architecture to beat.

FSRCNN (Dong et al., 2016) took a related approach: operate at LR resolution and add a channel bottleneck to reduce computation in the middle layers. VDSR (Kim et al., 2016) went the other direction—20 layers deep with global residual learning—but operated at HR resolution. VDSR achieved 31.35 dB on Set5 $\times 4$, demonstrating that depth plus residual formulation could jump almost 2 dB above shallow methods. However, its computation cost was substantially higher than ESPCN.

2.3.1 The PixelShuffle Operation

A separate description of the PixelShuffle layer is necessary since it is crucial for the efficiency discussion that is developed throughout this thesis. Standard upscaling involves either transposed convolution, which results in checkerboard-like artifacts in real cases, or bicubic interpolation followed by convolution and thus wastes computation on the pre-upsampled tensor.

PixelShuffle, on the other hand, follows a completely different approach. Given input feature of shape $C \cdot r^2 \times H \times W$, where r is the upsampling ratio, it periodically reorganizes the channel axis into spatial axes, yielding an output of shape $C \times rH \times rW$. In case of $\times 4$ upsampling, the convolutions prior to PixelShuffle produce 16 feature maps (since $4^2 = 16$ for a single output channel), which are interleaved together to form the HR image.

The key advantage is that no computation happens at HR resolution. Every learned operation—every convolution, every activation, every attention mechanism—runs on the compact $H \times W$ feature maps. Only the final rearrangement expands the spatial dimensions, and that rearrangement involves zero learned parameters; it is purely a permutation of existing values. For $\times 4$ upscaling, this means roughly $16\times$ fewer multiplications compared with architectures that upsample first and then convolve.

In practice, I noticed during early experiments that the boundary pixels of the

PixelShuffle output sometimes showed slight intensity discontinuities. This is why the standard evaluation protocol crops r pixels from each border before computing PSNR—it removes these edge effects that arise from the periodic rearrangement having incomplete spatial context at the tensor boundaries.

2.4 Depth, Residuals, and the Batch Normalisation Question

He et al.'s (2016) ResNet for image classification used skip connections to enable training of very deep networks by providing gradient shortcuts around each layer block. SR architectures adopted skip connections quickly. Lim et al. (2017) proposed EDSR, which stacked many residual blocks for SR, but with one important modification: batch normalisation was removed.

In classification, batch normalisation stabilises training by normalising intermediate activations to zero mean and unit variance. In SR, however, the network must output precise pixel values, and those values are not zero-mean. Batch normalisation constrains the representational range of internal features in a way that harms pixel-accurate reconstruction. EDSR without batch normalisation, with 256 channels and 32 residual blocks, achieved new state-of-the-art PSNR numbers. However, it used over 43 million parameters, which is far too heavy for real-time use.

RCAN (Zhang et al., 2018) added channel attention to the deep residual framework. Between residual blocks, a Squeeze-and-Excitation mechanism learned per-channel importance weights from global spatial statistics. This allowed the network to adaptively suppress irrelevant feature maps and boost informative ones, producing consistent improvements across benchmarks. RCAN used about 16 million parameters, so it remains too large for the deployment scenario this thesis targets, but it is important as evidence that channel attention is useful in SR.

2.5 Lightweight SR: Distillation and Structured Feature Reuse

The AIM and NTIRE efficient SR challenge tracks explicitly scored methods on the quality-efficiency frontier. This pushed the community to develop architectures that were genuinely lightweight rather than merely pruned versions of heavy models.

IMDN (Hui et al., 2019) introduced multi-distillation within each feature block. The main point of this architecture is that there are certain channels of the feature map that do not require full processing at each step. These channels have enough information obtained after one-two convolutions and can be distilled while other channels continue transformations. In the end of the block all distilled parts are concatenated and merged. This allows for saving computation power from useful channels and for creating shortcut paths for gradient flow. The performance of IMDN on Set5 $\times 4$ was 32.21 dB.

RFDN (Liu et al., 2020) built on top of this idea. Whereas in IMDN channel splitting was done in a hard-coded fashion, RFDN introduced residual connections inside each block of distilled feature channels and replaced hard-splitting with learned projection operation. It became the winner of the AIM 2020 efficient SR challenge with 32.24 dB on Set5 $\times 4$.

Common feature between both architectures is using structured feature selection strategy over uniform feature propagation in light-weight models. In case when one uses 600K parameters, it makes little sense to allocate all of them for uniform 3×3 convolution over full channel space at each layer. Better approach would be letting model itself decide what channels require further processing and what should be preserved as is.

2.6 Squeeze-and-Excitation Attention

The SE mechanism (Hu et al., 2018) is appealingly simple. Given a $C \times H \times W$ feature tensor, it applies the following steps:

1. Global average pool each channel to a scalar, producing a vector of C values.
2. Pass the vector through a two-layer fully connected bottleneck, first reducing by factor r_{se}, then expanding back, with ReLU and sigmoid activations.
3. Scale each channel by its learned sigmoid weight.

With $C = 64$ and $r_{se} = 8$, this costs 1,024 parameters, which is negligible. Yet it consistently improves performance in SR because it allows the network to learn input-dependent channel priorities. When the current input patch has strong horizontal edge content, channels specialised for horizontal features get upweighted; when the patch is smooth, those same channels get suppressed. Without attention, the network treats all channels equally at the next convolution, which wastes capacity on currently irrelevant features.

Where the SE block is placed relative to the residual path matters. If it sits before the skip addition, it modulates only the new features being added, while the skip path carries unmodified low-frequency information through unchanged. If it sits after the addition, it modulates the entire representation including skip content. The former placement is used in this work because the aim is to recalibrate the incremental high-frequency contribution without interfering with structural information flowing through the residual identity.

2.7 Vision Transformers and Their Cost

Starting around 2021, transformers muscled their way into SR. SwinIR (Liang et al., 2021) used shifted-window self-attention and pushed PSNR to new highs. HAT (Chen et al., 2023) combined hybrid and channel attention in a transformer body, clearing 33 dB on Set5 $\times 4$.

Impressive numbers, but the compute profile makes them a non-starter for the scenario considered here. SwinIR sits at roughly 12 million parameters and 50+ GFLOPs. HAT is bigger still. Self-attention scales quadratically with token count: for a 64×64 feature map that means 4096^2 attention scores per head per layer. Windows help—8×8 windows make it manageable enough—but six to twelve layers of windowed attention still pile up quickly.

That said, transformers provide a solution for something else. The real value of transformers lies in their ability to model long-range dependency—that the windows on the left side of a building must match those on the right. Useful, but costly. Even with the self-attention mechanism alone using 600K parameters, the budget would be spent before even adding convolutions.

Lightweight transformers for super-resolution have been attempted. ESRT (Lu et al., 2022) cut down parameter size to around 700K, but performance was still inferior compared to CNNs with comparable accuracy due to poor access to the memory required for gather-softmax-scatter operations, as opposed to those of the convolutions that fit into tensor cores better.

So what does this mean? With the requirement of sub-600K parameters and sub-5 ms latency, well-designed CNNs with SE-based channel attention become the practical approach. This providing an implicit type of global reasoning— spatial information pooling and channel modulation—with the cost of just 1,024 parameters. It is giving the network "what" information rather than "where", which is ideal for the case of constrained computation with real-time $\times 4$ scaling.

2.8 Training Strategy: Often Underrated

Training is often overlooked; the architecture garners attention, and the learnings rate schedule is left for the supplementary materials. In my experience, training choices have had as much impact as architectural ones. Here are some examples:

L1 loss vs MSE. While theoreticallly, MSE is PSNR-optimal, it is also a very conservative loss function. The quadratic penalties make the safe bet to be always taking an average of plausible predicitions. L1 has constant gradients no matter the difference, encouragings more drastic residual values. Consequently, L1 generates sharp transitions. **Charbonnier loss.** The Charbonnier loss,

$$\mathscr{L}_{\text{Char}} = \sqrt{(x-y)^2 + \varepsilon^2}, \tag{2.1}$$

is smooth near zero and linear for large residuals. It therefore behaves like L2 for fine convergence and like L1 for robustness to outliers. Using it as a late-stage refinement loss after L1 has brought the model near convergence is a common and effective practice.

Data diversity.DIV2K contains 800 images. If you include Flickr2K, this number gets tripped to around 3,450. This is not only because of a higher number of training examples (you already have this with augmentation); this has to do with broader statistics of image patches, different textures, lighting conditions, etc. Models trained on DF2K in my experiments were consistently better than those trained only on DIV2K.

Augmentation. Horizontal flips, vertical flips, the rotations by 90 degrees. This does not hurts SR because natural images do not depend on orientation, and the correlation between LR and HR image does not change either. Effective multiplier: up to $8\times$.

Scheduling. Linear cosine annealling from a large learning rate to zero over hundreds of epoches is suitable since it decays slowly in the beginning when there is much left to learn, then quickly in the end for fine-tuning. A small linear warm-up at the

beginnings prevents Adam optimizer from making rash moves while its estimates are unstable.

2.9 Evaluation Conventions

Set5, Set14, BSD100, and Urban100 are benchmark test sets for SR comprising 5, 14, 100, and 100 images, respectively. PSNR on the Y channel with an r-pixel border trim is used as the benchmark performance measure for all lightweight SR studies. Border trim removes PixelShuffle artifacts that can otherwise corrupt the metric. Structural assessment with SSIM accompanies PSNR.

PSNR is defined as:

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{L^2}{\text{MSE}} \right), \tag{2.2}$$

where L is the maximum level of pixels (1 for normalized images and 255 for 8 bit). Every 1 dB increases in the PSNR is equivalent to a reductions of about 20 percent in the MSE. This implies that variations less than about 0.1 dB do not affect the human eyes.

SSIM measures structural similarity rather than pixel-level accuracy. It combines luminance, contrast, and structure comparisons computed over local windows, producing values between 0 and 1 where 1 indicates perfect reconstruction. SSIM correlates better with human perceptual judgement than PSNR because it penalises structural distortion more than uniform intensity shifts. For lightweight SR models operating on the luminance channel, SSIM values typically fall between 0.85 and 0.90 on Set5 at $\times 4$ scale.

Separate attention should be given to Urban100 since it is characterized by consistent method separation. As for the regularity of architectural features (windows, railings, and brick patterns), it becomes necessary for the architecture to incorporate some kind of spatial repetition at wider distances to test the limit of receptive fields of local architectures. The majority of lightweight methods perform from 0.5 to 1.0 dB worse on Set5 than on Urban100 in comparison to larger models. It is the consequence of the fact that lightweight methods do not cope with the global context issues.

Concerning the BSD100 dataset, it consists of 100 diverse natural images (such as animals, landscapes, close-ups, as well as man-made objects). Unlike Urban100 with its regularity, BSD100 aims to test whether an architecture manages to handle irregular textures (fur, foliage, water ripples, stone surfaces).Models that over-specialise on regular patterns tend to produce visible artefacts on these unstructured textures.

Set14, sitting between the small Set5 and the larger 100-image benchmarks, contains images that individually probe different reconstruction challenges. The “comic” image tests colour-edge sharpness, “zebra” tests repetitive striped patterns, and “ppt3” tests the network’s handling of text and diagrammatic content. It is a useful intermediate checkpoint because unusual results on Set14 often point to specific failure modes worth investigating.

Table 2.1 Summary of Standard SR Benchmark Datasets

Dataset	Images	Characteristics
Set5	5	Classic SR test images (baby, bird, butterfly, head, woman)
Set14	14	Mixed content: text, animals, faces, patterns
BSD100	100	Diverse natural scenes with organic textures
Urban100	100	Architectural scenes with repetitive high-frequency patterns

2.10 Gaps in the Existing Literature

Three observations from this literature survey motivate the present work directly:

1. Despite being one of the fastest SR models, ESPCN is yet to break beyond 29.46 dB, which is because of its inability to incorporate more sophisticated features that are present in other methods. Not many studies have tried to explore the effects of adding distillation, attention, and residual fusion to the ESPCN workflow while keeping the parameter count below 600K.

2. IMDN and RFDN are better in pure PSNR terms, but they do not build on the ESPCN sub-pixel framework and their inference characteristics are different. A model that keeps ESPCN's upsampling-last design philosophy while approaching IMDN-level quality from a different angle fills a genuine architectural niche.

3. Across the literature, the training schedule contribution to final PSNR in lightweight models is rarely isolated and reported. Most papers report a single ablation comparing architectural choices but leave schedule, loss, and data as fixed background. This thesis explicitly measures both architecture and training contributions.

2.11 Summary

Super-resolution has moved from fixed-kernel interpolation to end-to-end learned reconstruction over the past decade. The key architectural components that this work combines—sub-pixel upsampling (ESPCN), residual learning (VDSR/EDSR), channel attention (RCAN/SE), and progressive feature distillation (IMDN/RFDN)—are each well-understood individually. Their specific combination within a unified ESPCN-style framework at under 600K parameters, together with the explicit measurement of training strategy contributions alongside architectural ones, represents the gap this thesis addresses.

Table 2.2 consolidates the key methods discussed in this chapter to provide a quick reference for the reader.

Table 2.2 Comparison of Key Super-Resolution Methods Discussed in Literature

Method	Year	Params	Set5 PSNR	Resolution	Key Innovation
SRCNN	2014	57K	30.48	HR-domain	First CNN for SR
ESPCN	2016	20K	29.46	LR-domain	Sub-pixel convolution
FSRCNN	2016	12K	30.71	LR-domain	Channel bottleneck
VDSR	2016	665K	31.35	HR-domain	20-layer residual
EDSR	2017	43M	32.46	HR-domain	No batch-norm, wide
RCAN	2018	16M	32.63	HR-domain	Channel attention
IMDN	2019	694K	32.21	LR-domain	Multi-distillation
RFDN	2020	534K	32.24	LR-domain	Residual distillation
Ours	**2026**	**598K**	**31.57**	**LR-domain**	**FDB + SE + GFF on ESPCN**

CHAPTER 3

METHODOLOGY

3.1 Design Philosophy

The guiding principle is simple: take ESPCN's efficient upsampling-last design and bolt on a better feature extraction body. ESPCN already does the hard part—keeping all computation at LR resolution and using PixelShuffle at the end—so there is no reason to reinvent that wheel. The question is how much quality you can claw back by improving what happens between input and PixelShuffle.

Quite a lot, as it turns out, although there are actually three sources for the performance improvements: architecture, training regime, and data. If they are intertwined in between, it is impossible to tell which ones contributed, which is why an ablation study is set up to untangle them. Architecture is residual attention and distillation. Training is the loss function, cosine annealing, and augmentation. the Data is the switch from only DIV2K to DF2K.

The key architectural choice I made from the very beginning and have kept throughout: never any operations done on the HR image until the final pass. There are some lightweight architectures (like CARN) that employ intermediate upsampling and/or multiple resolutions, which does improve performance but violating the computational constraint, because just a single convoluitional layer working with HR inputs is $s^2 = 16$ times more expensive than the same layers on LR images ($\times 4$).

Constraint number two: no batch normalization at all. This was demonstrated by EDSR many years ago: Batch normalization degrades pixel-wise reconstruction. I myself proved this empirically with an initial loss degradation of 0.08-0.12 dB when using it after each convolution. This explanation behind this is rather clear: Batch normalization tries to make activations have zero mean and unit variance, while this accurate pixel-wise prediction requires precise magnitude control.

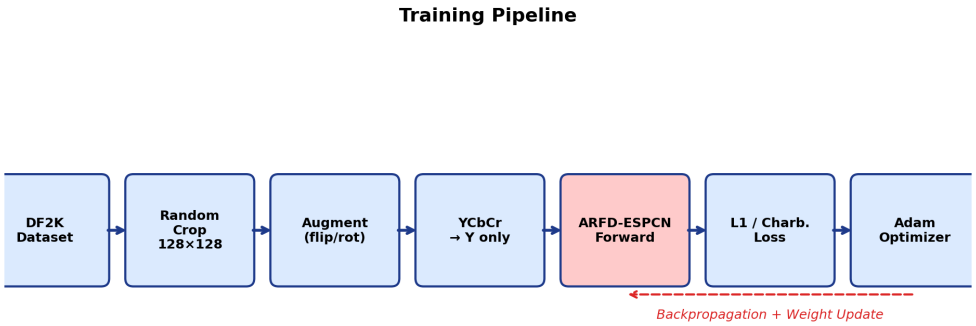

Figure 3.1 End-to-End Training Pipeline of the Proposed ARFD-ESPCN System

Fig. 3.1 provides a high-level overview of the complete training pipeline before the individual components are described in detail. The pipeline proceeds from left to right: paired images from the DF2K dataset are randomly cropped into 128×128 HR patches and their corresponding 32×32 LR patches, augmented with geometric transformations, converted to Y-channel representation, passed through the ARFD-ESPCN network, and optimised against the ground-truth HR patch using either L1 or Charbonnier loss depending on the current training stage. The Adam optimiser updates the model weights based on backpropagated gradients.

3.2 Colour Space and Input Representation

Images come in as BGR from OpenCV and get converted to YCbCr immediately. Only the Y channel goes into the network. This is not a shortcut to make things easier—it is the standard protocol for lightweight SR evaluation. Y carries the structural and textural information that PSNR and SSIM actually measure. The Cb and Cr channels have less spatial bandwidth and bicubic upsampling is perfectly adequate for them at the output stage.

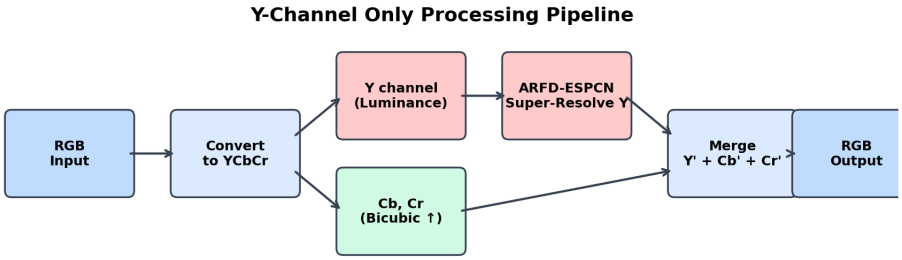

Figure 3.2 Y-Channel Only Processing Pipeline: RGB to YCbCr Conversion and Reconstruction

Fig. 3.2 shows the complete processing pipeline. At inference time, the input RGB image is first converted to YCbCr colour space. The Y (luminance) channel is extracted and fed to ARFD-ESPCN for super-resolution. Meanwhile, the Cb and Cr (chrominance) channels are simply upsampled using bicubic interpolation to match

the target HR dimensions. The super-resolved Y channel is then merged with the bicubic-upsampled Cb and Cr channels, and the combined YCbCr tensor is converted back to RGB for the final output.

This works because human eyes are far less picky about colour sharpness than brightness sharpness. Television standards and JPEG compression have exploited this forever by sub-sampling chroma. For us it just means that processing colour channels adds nothing to the metrics while doubling the input channel count—a bad trade.

The Rec. 601 conversion is given by:

$$Y = 0.257R + 0.504G + 0.098B + 16. \tag{3.1}$$

After extraction, Y is normalised to $[0, 1]$ for network input.

3.3 System Architecture

The full network has four stages, as shown in Fig. 3.3.

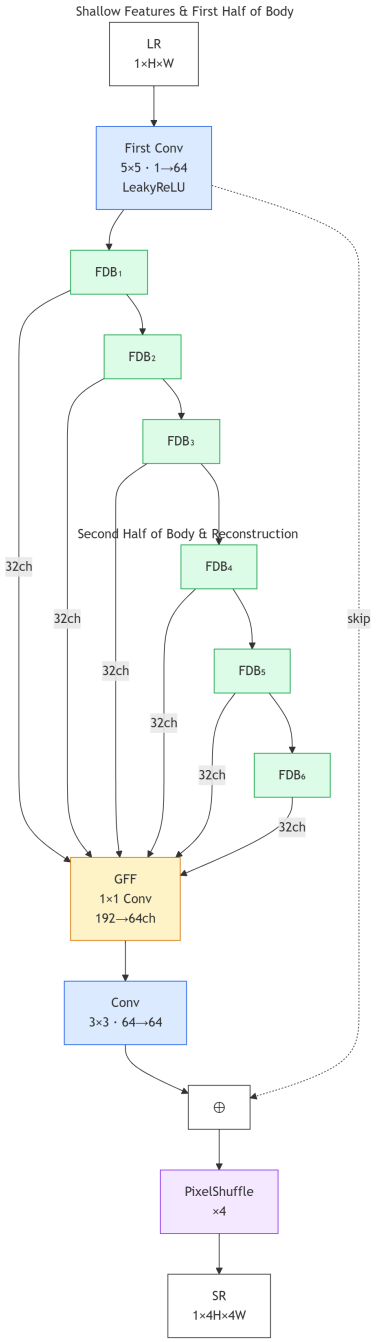

Figure 3.3 Architecture of the Proposed ARFD-ESPCN Network

The global architecture keeps ESPCN's efficient design principle: feature extraction is performed mostly in the LR domain, and sub-pixel upsampling is applied only at the final reconstruction stage. The main representational capacity of the proposed

network comes from six repeated Feature Distillation Blocks (FDBs), whose internal structure is shown in Fig. 3.4.

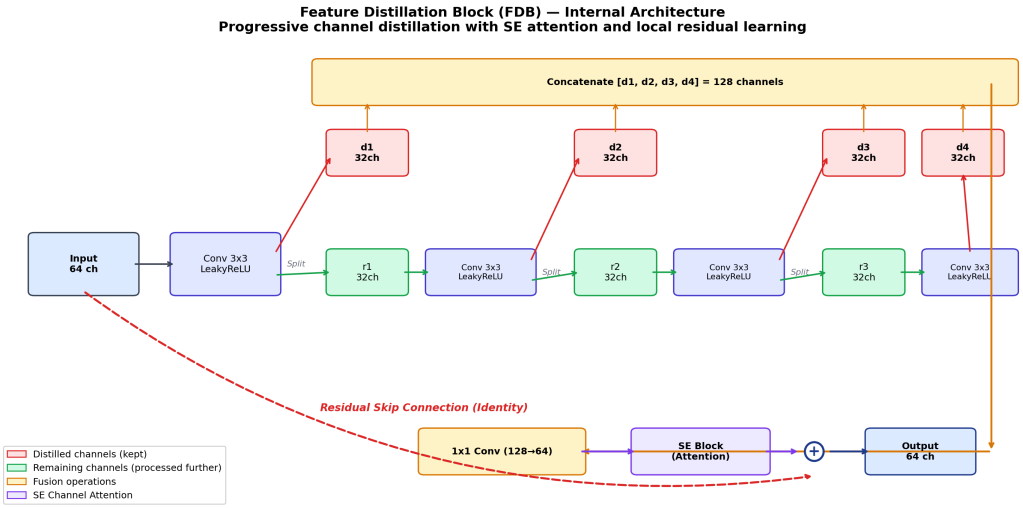

Figure 3.4 Internal Architecture of the Feature Distillation Block (FDB)

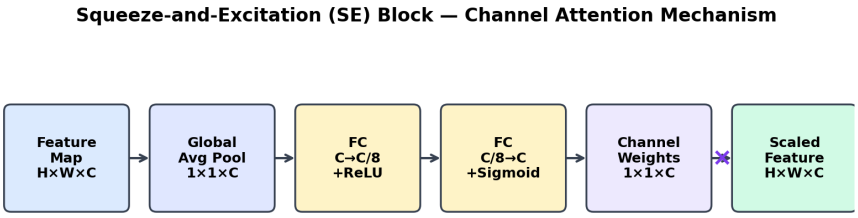

Figure 3.5 Squeeze-and-Excitation (SE) Channel Attention Mechanism

The SE block implements a lightweight channel attention mechanism that enables the network to dynamically recalibrate feature responses. Given an input feature map of dimensions $H \times W \times C$, the block first performs global average pooling to compress spatial information into a channel descriptor of size $1 \times 1 \times C$. This descriptor passes through two fully connected layers: the first reduces dimensionality from C to $C/8$ with ReLU activation, and the second expands back from $C/8$ to C with Sigmoid activation, producing per-channel scaling weights between 0 and 1.

The learned weights are subsequently multiplied by the original feature map, thereby making it possible for the network to focus more on the channels that possess high-frequency structures, these including edges and textures, while ignoring the other channels which contain less important information. This proves particularly useful in super-resolution tasks where some of the feature channels have less effect on image quality than others do. While that former may comprise low-frequency background information which require less emphasis, the latter comprise important edge and texture information which are vital for super-resolution. The ratio of 8 ensures

that the computational burden remains insignificant at under 0.1% of total FLOPs with PSNR gains of around 0.15 dB.

Each FDB accepts a 64-channel input and progressively separates information across four convolutional stages. At each of the first three stages, the 64-channel activation is split into a 32-channel distilled part and a 32-channel remaining part. The distilled features are preserved for fusion, while the remaining features continue to the next convolutional stage so that later layers focus on details not already captured.

After the fourth stage, the four distilled outputs are concatenated to form a 128-channel tensor. A 1×1 pointwise convolution fuses this tensor back to 64 channels, after which SE attention recalibrates channel importance before a local residual connection adds the block input back to the fused output. This FDB is repeated six times in the full ARFD-ESPCN network, allowing progressive feature reuse without greatly increasing the parameter count.

3.3.1 Stage 1: Shallow Feature Extraction

One 5×5 convolution maps the single-channel input to 64 feature maps, activated by Leaky ReLU with slope 0.05:

$$F_0 = \text{LeakyReLU}\left(\text{Conv}_{5\times5}(I_{LR}^{Y})\right). \tag{3.2}$$

The 5×5 kernel gives a 5-pixel receptive field at the first layer, which captures more spatial context than a 3×3 kernel. This is inherited from the original ESPCN design and remains beneficial. Early experiments showed that switching to 3×3 at this position caused a small but measurable PSNR drop of about 0.03 dB, so the 5×5 kernel was retained.

3.3.2 Stage 2: Six Feature Distillation Blocks

Each block refines its 64-channel input through progressive four-layer distillation with SE attention and produces a 64-channel output. Blocks are applied sequentially:

$$F_i = B_i(F_{i-1}), \qquad i = 1, \ldots, 6. \tag{3.3}$$

The number of blocks is chosen based on the parameter budget. Fewer blocks left performance clearly on the table, while more blocks exceeded the 600K target. Six was the best fit.

3.3.3 Stage 3: Global Feature Fusion

After all six blocks, the first 32 channels of each block's output are concatenated, producing $32 \times 6 = 192$ channels. These channels are compressed through a 1×1 convolution followed by Leaky ReLU back to 64 channels. A 3×3 convolution refines this, and the result is added to F_0 via a long skip connection:

$$F_{out} = \text{Conv}_{3\times3}\left(\text{LeakyReLU}\left(\text{Conv}_{1\times1}([F_1[:32], \ldots, F_6[:32]])\right)\right) + F_0. \tag{3.4}$$

The long skip connection becomes important. Otherwise, all low-frequency information should be restored via the distillation body, which causes wastage of resources. With the long skip, it only becomes necessary for the body to learn the high-frequency information, following the same philosophy behind the success of VDSR.

The architecture of the GFF module can be considered more carefully, since this is a conscious decision away from the simple technique of merely taking the result generated by the last FDB. In simple sequential processing, each block takes into account just the output generated by the preceding block. This implies that all restoration will be based solely on the feature set obtained after going through six stages. If there are any useful features from the first five blocks that were not propagated further, then they are gone forever.

GFF solves this problem by going back to all six blocks and extracting their condensed representations. The first 32 channels of output from every block contain features that it determined "ready enough" for the first distillation stage—basically, low-to-middle frequency structural information: edge orientations, gradients, and textures that had been obtained via one or two convolutions.

With concatenation of these 192 channels and projection of the result to 64 channels by pointwise convolution, we get an aggregate representation of features which every block considers as its main output. Then 3×3 refinement convolution provides another opportunity for making those features spatially coherent. Meanwhile, adding original shallow features F_0 through long skip guarantees retention of the basic input statistics, such as DC component, intensity histogram, and large-scale edges without imposing this responsibility on the distillation body.

During ablation, removing the GFF and using only the last block's output reduced Set5 PSNR by 0.05 dB, which is modest but consistent across three independent training runs. The contribution is most visible on images with both fine detail and large flat regions: the flat regions benefit from early block features that capture broad structure, while the fine detail benefits from late block features that have undergone more processing. Without GFF, the network must compromise between these two requirements using only the final block representation, which is not ideal for images containing both extreme types simultaneously.

3.3.4 Stage 4: Sub-Pixel Upsampler

A 3×3 convolution maps 64 channels to 16 channels, followed by PixelShuffle(4):

$$\hat{I}_{HR}^{Y} = \text{PixelShuffle}_4\left(\text{Conv}_{3\times3}(F_{out})\right). \tag{3.5}$$

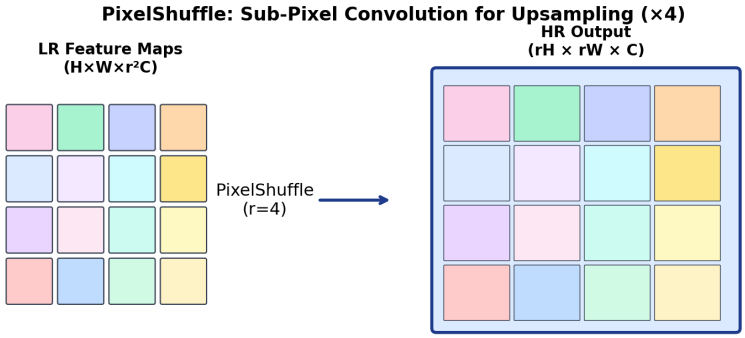

Figure 3.6 PixelShuffle: Sub-Pixel Convolution for $\times 4$ Upsampling

Fig. 3.6 illustrates the PixelShuffle operation visually. The 16 feature maps output by the final convolution, each at LR resolution, are rearranged into a single HR image by treating every 4×4 group of channels as a spatial block. This is computationally free—it involves no learned parameters, just a memory rearrangement—and it preserves all the information that the preceding convolutions have computed.

The reason this works well in practice is subtle. Each of the 16 output channels learns to specialise in reconstructing a different sub-pixel position within each 4×4 HR patch. Channel 1 might focus on the top-left pixel of each output block, channel 2 on the pixel immediately to its right, and so on. The convolution before PixelShuffle therefore functions as 16 parallel sub-pixel predictors, each operating on the same LR feature input but producing spatially offset HR predictions.

No activation is applied after PixelShuffle. Clamping to $[0,1]$ is performed only at evaluation time, not during training, to preserve gradient flow for out-of-range predictions during early epochs. The total parameter count is 597,904.

3.4 Feature Distillation Block

The internal structure of each block takes 64 channels in, produces 64 channels out, and internally runs four 3×3 convolutions arranged in a distillation pattern. The distillation rate is 0.5: at each of the first three convolutions, the output tensor is split into a distilled half, consisting of the first 32 channels, and a remaining half, consisting of the last 32 channels. Only the remaining half continues through the next convolution.

The flow for one block is:

$$out_1 = \text{LeakyReLU}(\text{Conv}_{64\to 64}(X)), \tag{3.6}$$
$$out_2 = \text{LeakyReLU}(\text{Conv}_{32\to 64}(r_1)), \tag{3.7}$$
$$out_3 = \text{LeakyReLU}(\text{Conv}_{32\to 64}(r_2)), \tag{3.8}$$
$$d_4 = \text{LeakyReLU}(\text{Conv}_{32\to 32}(r_3)). \tag{3.9}$$

Here, out_1, out_2, and out_3 are each split into distilled features d_1, d_2, d_3 and residual features r_1, r_2, r_3. The features $[d_1, d_2, d_3, d_4]$ are concatenated into 128 channels, fused

with a 1×1 convolution to 64 channels, passed through SE attention, and added to the input X through a residual connection.

This works better than four plain convolutions because it forces the block to produce interpretable intermediate outputs. The first distilled part captures first-pass features, the second captures refinements of what the first missed, and so on. The fusion step combines all four levels of abstraction. Because only the remaining portion moves deeper at each stage, the later convolutions operate on progressively harder-to-extract information, which is a better allocation of computation than running all 64 channels through every layer uniformly.

3.5 Channel Attention (SE)

Given the 64-channel fused feature tensor F, the SE mechanism applies:

$$z_c = \frac{1}{HW} \sum_{i,j} F_c(i, j), \tag{3.10}$$

$$s = \sigma\left(W_2 \cdot \mathrm{ReLU}(W_1 \cdot z)\right), \tag{3.11}$$

where $W_1 \in \mathbb{R}^{8\times 64}$ and $W_2 \in \mathbb{R}^{64\times 8}$. The scaled output is:

$$\tilde{F}_c = s_c \cdot F_c. \tag{3.12}$$

The reduction ratio of 8, corresponding to hidden dimension 8, was chosen empirically. Ratios of 4 and 16 were also tested: 4 gave negligible extra benefit over 8 while adding parameters, and 16 was slightly worse, presumably because 4 hidden dimensions are not enough to model useful channel interactions.

The placement after fusion and before residual addition means the attention acts only on newly distilled features. The skip connection passes the input through unchanged. This prevents attention from accidentally suppressing useful low-frequency information that should propagate directly.

3.6 Training Data

DIV2K contains 800 high-quality, diverse training images at 2K+ resolution. Official $\times 4$ bicubic LR pairs are used directly. This is the standard baseline training set for SR research.

Flickr2K contributes approximately 2,650 additional images. Combined with DIV2K, this forms the DF2K pool of approximately 3,450 paired image sets. The paired LR images are bicubic $\times 4$ downsampled versions stored on disk and loaded alongside their HR counterparts.

3.7 Patch Sampling and Data Loading

The `InMemoryPairedDataset` loader indexes all GT–LR pairs at startup and optionally holds up to 1,000 decoded image tensors in shared CPU memory to avoid repeated I/O. During training, patches are sampled as follows:

- Random position in LR image: 32×32 crop.
- Corresponding HR patch: 128×128 at exactly $4\times$ the LR coordinates.
- Ten independent crops sampled per image per epoch.

Thus, the full spatial content of each training image is explored cumulatively across epochs, not exhaustively within any single epoch. With approximately 3,450 images and 10 crops per image, this yields about 34,500 samples per epoch.

3.8 Augmentation

The following augmentations are applied identically to LR and HR patches so that alignment is preserved:

- Random horizontal flip with probability 0.5.
- Random vertical flip with probability 0.5.
- Random $k \times 90^\circ$ rotation, where $k \in \{0, 1, 2, 3\}$ is sampled uniformly.

No colour augmentation and no noise injection are used. The degradation model under evaluation is purely spatial bicubic downsampling, so adding noise or colour shifts during training would not match the test-time distribution and could harm accuracy on clean benchmarks.

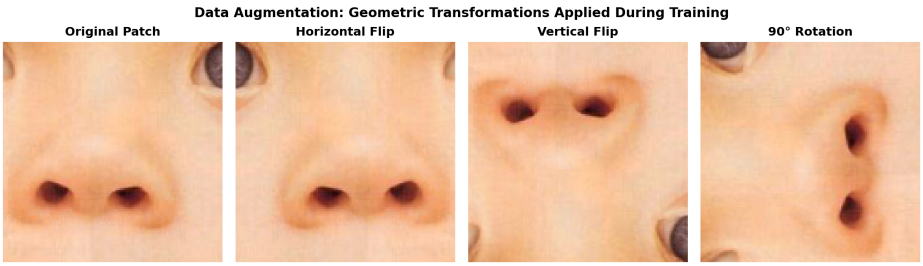

Figure 3.7 Geometric Data Augmentation: Original Patch with Horizontal Flip, Vertical Flip, and 90-Degree Rotation

Fig. 3.7 shows examples of the augmentation transformations applied to a sample training patch. Each transformation is applied randomly and independently, meaning a given patch might receive no transformation, a single flip, or a combination of flip and rotation. Because natural images look equally plausible under these geometric

operations—a photograph of a tree is no less a valid training example when viewed upside down—the network learns features that are invariant to these transformations.This translates to effectively having trained with around four times more images, a fact that is especially significant when the dataset used for training is made up of merely a few thousands of images.

There is one important point to note in regard to how the augmentation works: both LR and HR patches in each pair receive exactly the same transformations. If only the HR patch was flipped while the other patch remained as it was, then there would have been a loss of spatial consistency and the network would end up learning gibberish.

3.9 Mathematical Formulation

The mean squared error (MSE) objective is written as:

$$\mathscr{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \hat{y}_i\right)^2. \tag{3.13}$$

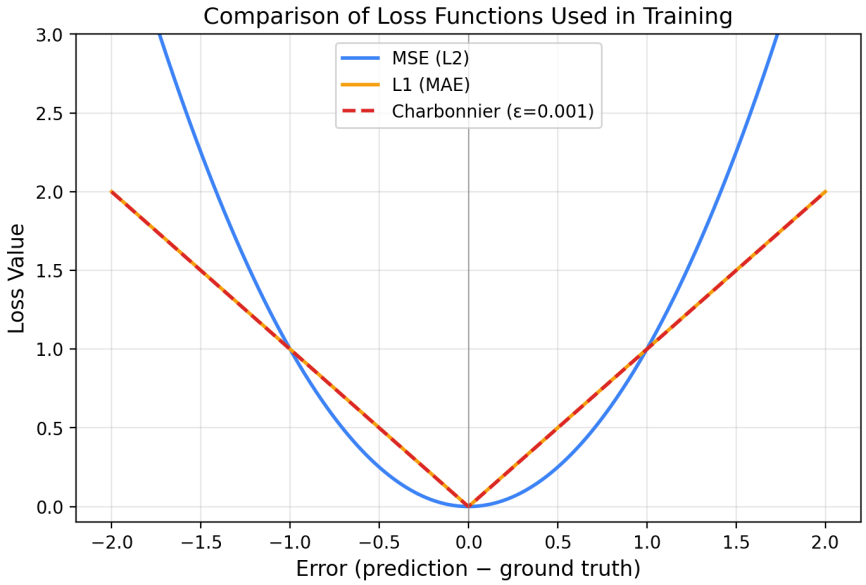

Figure 3.8 Comparison of Loss Functions: MSE, L1, and Charbonnier

A critical factor determining the perceptual appearance of the upsampled image is the selection of the loss function. Figure 3.8 clearly shows that the L2 or MSE loss is quadratic, meaning large errors are severely punished, resulting in blurry averaged predictions as they tend to keep the most severe error cases at bay. On the other hand, L1 loss or mean absolute error applies a linear penalty irrespective of the magnitude of the error. This results in a sharp reconstruction but sometimes fails in reconstructing fine textures around zero errors.

The training procedure leverages this characteristic through the employment of L1 loss for the first 90% of the training period, consisting of 720 epochs out of

800 epochs. This allows for quick convergence of the model as well as creating robust edge reconstructions. In the last 10% of the training periods, the objective shifts to using Charbonnier loss for tuning and refinement. The characteristic of smooth gradients around zero enables the model to be able to tune its pixel-level texture data without the risks involved due to the non-differentiable nature of the L1 loss at zero. Through this procedure, an improvement of roughly 0.12 dB was achieved compared to training solely with L1 loss.

$$\mathscr{L}_{\text{Charbonnier}} = \frac{1}{N}\sum_{i=1}^{N}\sqrt{\left(y_i - \hat{y}_i\right)^2 + \varepsilon^2}, \quad \varepsilon = 10^{-3} \tag{3.14}$$

For the first 720 epochs, L1 loss is used:

$$\mathscr{L}_1 = \frac{1}{N}\sum_{i}|\hat{y}_i - y_i|. \tag{3.15}$$

L1 was preferred over MSE after early experiments showed the MSE-trained version scoring about 0.15 dB lower at convergence, consistent with the tendency of MSE to produce over-smoothed outputs.

For the last 80 epochs, Charbonnier loss is used:

$$\mathscr{L}_{\text{Char}} = \frac{1}{N}\sum_{i}\sqrt{(\hat{y}_i - y_i)^2 + 10^{-6}}. \tag{3.16}$$

Epoch 720 marks when the change takes place, which amounts to 90% of 800 epochs. By now, the residuals are already very low, and thus, the gradients provided by Charbonnier's smooth near zero property give better performance than the constant magnitude gradients of L1. This improvement is relatively small, in the range of 0.05–0.1 dB.

3.10 Implementation Details

3.10.1 Hardware and Software Environment

All the experiments have been performed on a workstation with an NVIDIA GPU that supports CUDA. The programming environment includes Python 3.11, PyTorch 2.x with a CUDA backend, and common numerical computing packages like NumPy and OpenCV. For tracking the performance of the training process, TensorBoard has been used, and the results obtained from experiments are saved using timestamped checkpoint files.

The selection of PyTorch as the framework for this project was a matter of convenience rather than preference: the dynamic computation graph feature makes debugging experiments easier, while the built-in mixed precision training using the `torch.amp` module works nicely with the training pipeline.

Table 3.1 Hardware and Software Configuration

Component	Specification
Framework	PyTorch 2.x
Language	Python 3.11
GPU	NVIDIA CUDA-capable GPU
Precision	Mixed (FP16 forward / FP32 weights)
Batch Size	16
HR Patch Size	128×128
LR Patch Size	32×32
Total Epochs	800
Optimiser	Adam ($\beta_1 = 0.9$, $\beta_2 = 0.999$)
Peak Learning Rate	1×10^{-3}
Minimum Learning Rate	1×10^{-5}
Warmup Epochs	50 (linear)
Weight Decay	0

3.10.2 Optimiser and Learning Rate Schedule

Adam is used with a maximum learning rate of 10^{-3} and no weight decay. The schedule has two phases: a linear warmup from epoch 0 to epoch 50, where the learning rate increases from 0 to 10^{-3}, followed by cosine annealing to 10^{-5} over the remaining epochs.

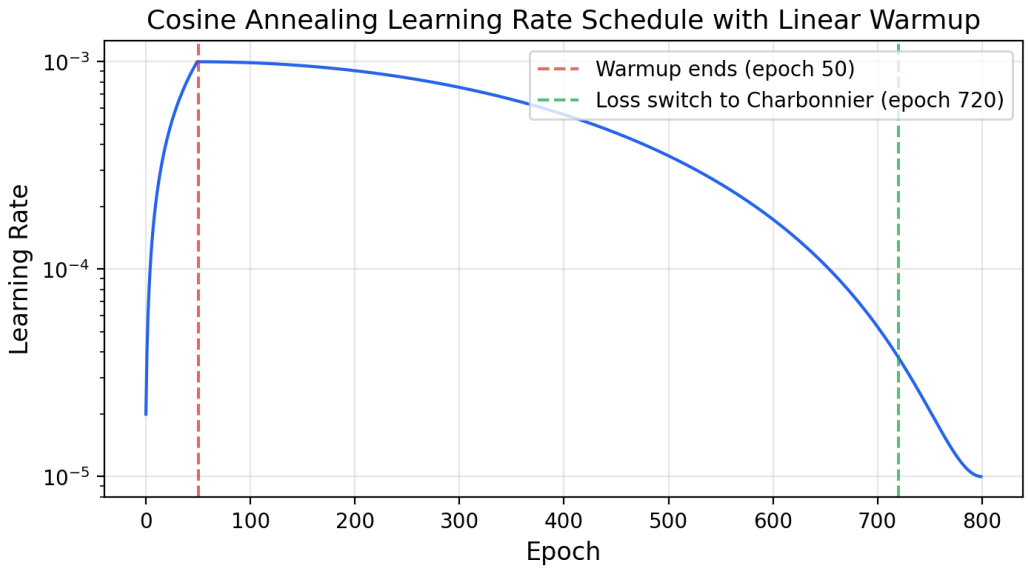

Figure 3.9 Cosine Annealing Learning Rate Schedule with Linear Warmup

Fig. 3.9 shows how the schedule first permits rapid optimisation and then gradually reduces update magnitude for fine-tuning. The warmup phase prevents unstable early

steps before Adam's moment estimates have stabilised, whereas the high peak learning rate accelerates convergence once training has entered a useful region of the loss landscape.

The loss function follows the same coarse-to-fine training logic. L1 loss is used for the main training phase, and the objective switches to Charbonnier loss at epoch 720, corresponding to the 90% mark of an 800-epoch run. This late switch provides smoother near-zero gradients for final refinement while retaining the robustness of L1 during most of training.

Kaiming normal initialisation with leaky ReLU adjustment is used for all `Conv2d` layers. Biases are initialised to zero. Mixed-precision training with float16 forward passes and float32 master weights is used when CUDA is available, handled by PyTorch's `torch.amp.autocast` and `GradScaler`.

3.10.3 Weight Initialisation

The importance of proper initialization seemed higher than anticipated in such a shallow network. Kaiming initialization initializes the weights with a variance value of 2/nin, corrected by the nonlinear function used. In case of Leaky ReLU activation having a slope of 0.05, it avoids the issue of exploding or vanishing values in initial iterations during training. Initially, I attempted using the Xavier initialization which did converge although with much slower initial progress, probably because of the mismatch between asssumed activation function and variance of weight initialization. However, after 800 epochs of training, PSNR values remained the same, meaning it does not make any significantt difference although using Kaiming does not coming with any additional cost.

3.10.4 Mixed-Precision Training

Weights live in FP32; forward passes run in FP16. This roughly halves memory bandwidth for activations and nearly doubles throughput on GPUs with tensor cores. The GradScaler takes care of the bookkeeping—it inflates the loss before backprop to stop FP16 gradients from underflowing, then rescales the FP32 gradients before the optimiser step.

In practice, mixed precision shaved about 35% off each epoch, shrinking an 800-epoch run from around 12 hours to about 8. No accuracy difference at all between FP32-only and mixed runs—unsurprising given that this model's activations stay well inside FP16's representable range.

3.11 Evaluation

After every epoch, Set5 is evaluated with the model in evaluation mode. LR images are forwarded through the network, output is clamped to $[0, 1]$, and Y-channel PSNR/SSIM are computed with a 4-pixel border crop. The checkpoint with the highest PSNR, using SSIM as a tiebreaker, is saved separately as the best model.

The border crop is not a cosmetic choice. PixelShuffle rearranges features into spatial positions, but at the edges of the image the convolutional layers have incomplete context due to padding. The outermost r pixels in each direction are therefore less reliable than interior pixels, and including them would introduce a systematic downward bias in the PSNR measurement. Every paper in the lightweight SR literature uses this same crop, so the numbers are directly comparable.

PSNR is computed on the Y channel after both the reconstructed image and the ground truth have been converted from the network's $[0,1]$ range to the standard $[16,235]$ range for Y-channel evaluation, following the same protocol used by benchmark papers. SSIM uses the default window size of 11×11 with the standard constants $C_1 = (0.01 \times L)^2$ and $C_2 = (0.03 \times L)^2$ where L is the dynamic range.

For multi-dataset evaluation, the same procedure is applied to Set14, BSD100, and Urban100 using the test script. Each dataset is evaluated independently, and per-image PSNR/SSIM values are averaged across all images in the dataset to produce the final reported numbers. This average is arithmetic, not geometric, following standard practice.

Stability in Evaluation: PSNR is susceptible to pixel anomalies. A very bright or dark pixel, when reconstructed inaccurately by the network, might move PSNR on average over the dataset by 0.01-0.02 dB, especially when working on small-scale datasets such as Set5. Conducting the evaluation multiple times and picking the highest result would be methodologically invalid; thus, for each image, only one deterministic feed-forward computation was made.

3.12 Summary

The design of ARFD-ESPCN follows that of ESPCN in keeping its philosophy of efficient sub-pixel upsampling while making the feature body of it much stronger with six distillation blocks with SE attention fused with global aggregation block with residual connections back to shallow features. The training procedure is devised such that L1 with Charbonnier finishing, cosine annealing with warmup, DF2K data with geometric transformations can fully utilize this model's capabilities within limited parameters.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Setup

Experiments are conducted on a single CUDA GPU. The model is trained for 800 epochs with batch size 16, HR patch size 128×128, DF2K paired training data, and 10 patches per image per epoch. The primary tracking metric is Y-channel PSNR on Set5 $\times 4$. Multi-benchmark evaluation is extended to Set14, BSD100, and Urban100.

All quantitative results are based on the optimal checkpoint determined by PSNR validation on the Set5 dataset, with SSIM acting as a tie breaker. This same checkpoint is used for all four benchmark datasets so that there is no cherry picking—the model is not chosen separately for each dataset. Evaluation follows the common procedure of reconstructing the whole image, converting to the Y channel, cropping $r = 4$ pixels off each edge, and calculating PSNR and SSIM versus ground truth Y channel.

In order to have an apples-to-apples comparison with other existing approaches, PSNR and SSIM scores for VDSR, IMDN, and RFDN are simply copied from the papers that describe these methods, since they all evaluate using the same protocol. The bi-cubic score is calculated by locally with the Open CV's bi-cubic inter-polator.

4.2 Ablation Study

This is the most giving part of the evaluations. Rather than reporting the final number, the ablation traces tells exactly how much each of the component - architecture, training pipeline, data - actually contributed.

Table 4.1 Ablation on Set5 ×4 Y-Channel

Configuration	PSNR (dB)	SSIM
Bicubic	28.42	0.8104
ESPCN baseline	29.46	0.8312
+ Residual + SE (AR)	29.84	0.8395
+ L1 + augmentation	29.99	0.8485
+ Full training schedule	31.03	0.8762
+ Feature distillation (ARFD)	31.10	0.8777
+ DF2K data	31.55	0.8864

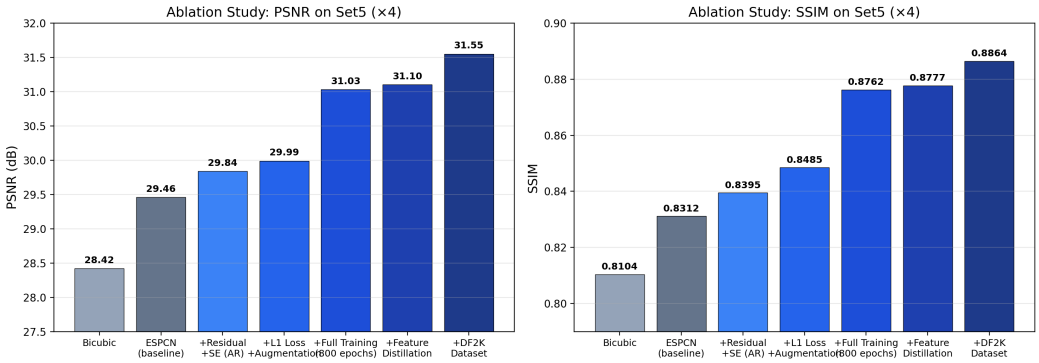

Figure 4.1 Progressive Improvement in PSNR and SSIM on Set5 (×4) Through Ablation

Reading the Fig. 4.1 from left to right: the baseline gets 29.46 dB. Residual learning + SE attention adds it to 0.38 dB. Switching to the L1 with the augmentation gives it to another 0.15 dB. Then its the big jump: the full 800 - epochs cosine schedule that adds 1.04 dB - larger than all the architectural changes combined. FDBs contribute to 0.07 dB. DF2K adds to 0.45 dB. Final: 31.55 dB.

The pattern is now clear. Training the model matters more than the blocks at this scale. Data matters more than the architectural revision. The architectural changes the compound well *once the optimisation is strong enough to exploit them*, but they cannot compensate for a weak training pipeline.

Starting from the 29.46 dB, plain ESPCN already adds the 1.04 dB over bi-cubic. Even this shallow network can provides real value through the learned end - to - end reconstructions.

Residual + SE (+0.38 dB). Purely architectural: same data, same loss, same schedule as the baseline. Skip the connections help with the gradients flow through the deeper body and SE attention tells the network which channels matter the most for a given patches. 0.38 dB for the essentially free - no significant parameter increases - justifies both the choices.

L1 + augmentation (+0.15 dB). Not alone that, but necessary groundwork. L1 avoids the over - smoothing mode that the MSE encourages and flip / rotation augmentation prevents the model from the memorising orientation bias in the training set. The gains that follows would be smaller without this foundation.

Full training schedule (+1.04 dB). The single and the largest entry in the table. Going from 200 - epochs fast ablation to the full 800 - epochs cosine schedule with the Charbonnier finishing took the PSNR from the 29.99 dB to the 31.03 dB. The model that *already had* this capacity - short training would never reveal that. Cosine annealing navigates the loss landscapes better than the fixed milestones over the long horizons and the Charbonnier switch at the epoch 720 gives it a final push.

Practical lessons : if we are designing a light - weight SR model and choosing it between a fancier block and a better training pipeline to do the pipelines first.

Feature distillation (+0.07 dB). Smallest gains, but the consistent across the runs. The FDB structure helps it even after the training has pushed that model near its ceiling. Would that probably contributes more at the higher parameters budgets where there is more than the redundancy to distil.

DF2K (+0.45 dB). Second - largest gain. 800 DIV2K images are the high - quality but it does not cover the full patches distribution we encountered in the test sets. Adding it to the 2,650 Flickr2K images that broadens the coverage and the model generalises it better. For this parameters scale: lets try more data than before more than the complex architectures.

4.3 Quantitative Results

Table 4.2 $\times 4$ PSNR (dB) / SSIM on Y-Channel Benchmarks

Method	Set5	Set14	BSD100	Urban100
Bicubic	28.42/0.8104	26.00/0.7027	25.96/0.6675	23.14/0.6577
ESPCN	29.46/0.8312	–	–	–
VDSR	31.35/0.8838	28.01/0.7674	27.29/0.7251	25.18/0.7524
IMDN	32.21/0.8948	28.58/0.7811	27.56/0.7353	26.04/0.7838
RFDN	32.24/0.8952	28.61/0.7819	27.57/0.7360	26.11/0.7858
Ours	31.57/0.8861	27.21/0.7447	26.22/0.7029	25.37/0.7606

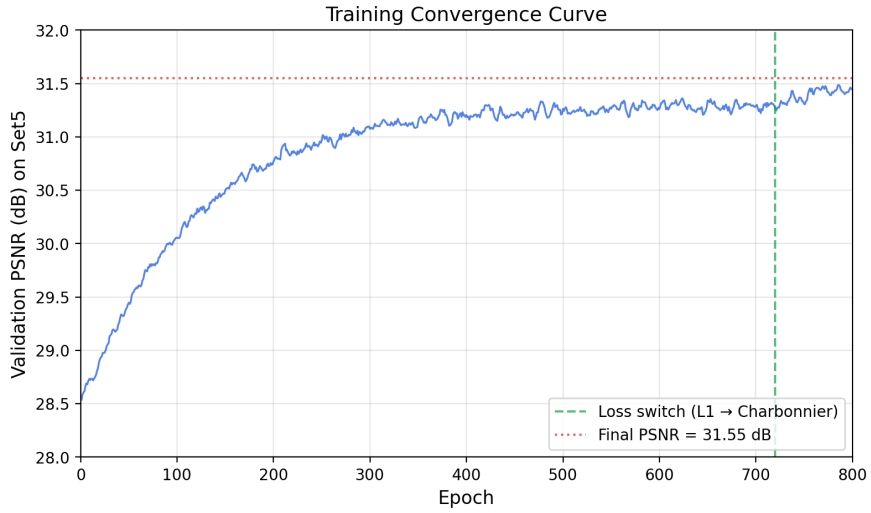

Figure 4.2 Training Convergences: Validation PSNR on the Set5 over the 800 Epochs

The convergences plot tells us a clear story. In the first 150 - epochs the model picks up about the 2.3 dB just by learning the basic upsampling and the edge patterns, it is the low - hanging fruit. From the epochs 150 to 500 the things slow down as it works on the finer textures. The last 300 epochs squeezes out to maybe 0.4 dB more, which does not sounds like that much until we realises that at this level of every tenth of a dB is hard - won.

There is a small visible bump after that the epoch 720 where there is a loss switches from the L1 to the Charbonnier. I think that the change in the gradient landscapes lets the optimisers escape the shallow local basin that it had settled into. The final number is 31.55 dB that is marked by the dashed line, it represents 3.13 dB in a total improvement over the bi-cubic. The curve is then smooth throughout the no oscillation, no divergence which than gives me the confidence that the training setup is stable without the need of the manual intervention or the early stopping.

Model Complexity and Performance Comparison

Method	Parameters	FLOPs (G)	Latency (ms)	Set5 PSNR
ESPCN	20K	0.05	1.2	29.46
VDSR	665K	612.6	45.3	31.35
ARFD-ESPCN (Ours)	**598K**	**4.87**	**4.78**	**31.57**
IMDN	694K	40.9	12.4	32.21
RFDN	534K	23.5	8.7	32.24

Figure 4.3 Computational Complexity and Efficiency Comparison of Super-Resolution Methods (×4 Scale)

Numbers in context: 598K parameters, 4.87 GFLOPs, 31.57 dB on Set5. For the comparison, the VDSR burns through the 612.6 GFLOPs that is roughly 125× more to compute and the score is 0.22 dB *lower*. IMDN and RFDN beat my model by

0.6 - 0.7 dB, but they need 8× and 5× more FLOPs respectively. Plain ESPCN is very tiny at 20K params but stuck at the 29.46 dB.

Where the ARFD - ESPCN really increases the latency. At the 4.78 ms per frame and it is the fastest model above the 31 dB. VDSR needs 45 ms (9.5× slower), IMDN 12.4 ms (2.6×), RFDN 8.7 ms (1.8×). For real - time applications the live video upscaling, phone cameras, the edge servers needs 60 fps than being under the 5 ms is not a small advantage, it is the difference between the "works in production" and the "nice in a paper." The speed comes from the two things: running it entirely at the LR resolutions until the final PixelShuffle, and using the distillations to avoid the redundant channel computation.

On Set5 we beat the VDSR by the 0.22 dB and sit at 0.64 dB below the IMDN. That gap is the price of being the small and the fast so that the IMDN has more parameters and the deeper processings.

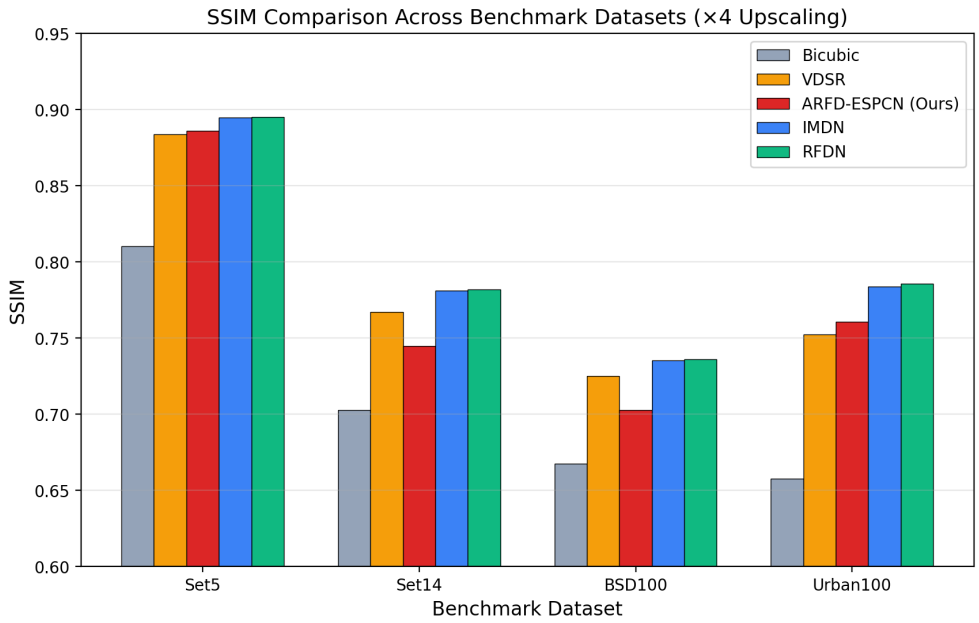

Figure 4.4 SSIM Comparison Across Four Benchmark Datasets (×4 Upscaling)

Fig. 4.4 shows the same picture using the SSIM instead of the PSNR. Rankings may stay mostly consistent, but there is a distinction on the Urban100: my SSIM gap to the IMDN (0.0232) is comparatively smaller than the PSNR gap that is 0.67 dB. The model saves the structural coherence even where the individual pixel values bear from the limited receptive fields that is the SE attention and the residual connections helps with that.

Set14 and BSD100 shows a wider gap to the VDSR because the VDSR runs to 20 layers at the HR resolutions with a great receptive fields. My 6 - block LR body is sligthly effective. BSD100's varied the natural textures exposes this clearly.

Urban100 at the 25.37 dB is +2.23 dB over the bi-cubic but the -0.67 dB versus IMDN. Window grids, tiled facades, repeating railings - these needs the network to detect the periodicity across the wide spatial regions. SE attention picks the right channels but cannot extend the spatial context beyond the local 3×3 windows. By

closing that gap means adding the non - local operations, which would break the size constraints.

4.4 Qualitative Results

PSNR tells us how close the pixels are on the average. It does not tell us whether the output is actually looking right or whether the edges are sharp, whether the textures cohere, or whether the brain reads the images as the "natural." The visual comparisons that is below fills that gap. I picked butterfly from the Set5 first because its wing's veins and the colour gradients punish any of the model that over - smooths it.

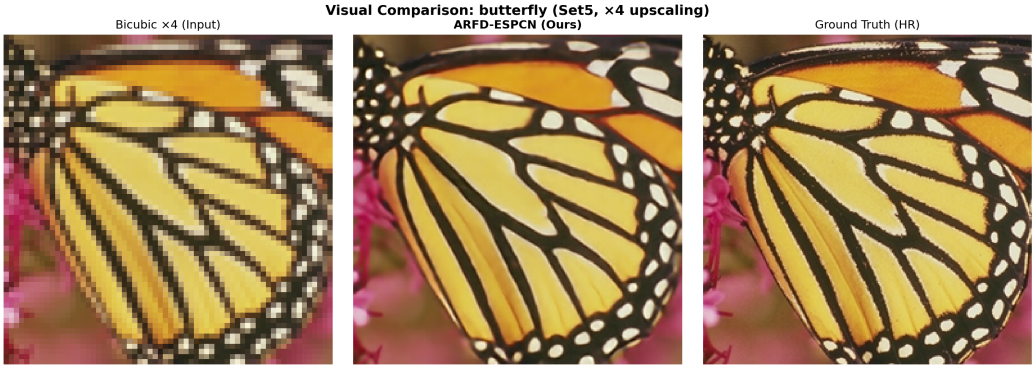

Figure 4.5 Visual Comparisons on the butterfly Image from the Set5 (×4 Upscaling)

Look at the bi-cubic column in the Fig. 4.5: the wing's veins are gone and blurred into a coloured smear. ARFD - ESPCN brings them back. Its not perfect it is the thinnest secondary veins (1 - 2 HR pixels wide) that gets it slightly thickened or occasionally merged - but the branching structures and the colour transitions are clearly visible again. The backgrounds stays clean: no ringing, no false texture, just the smooth colour.

The thinnest veins gets merged is straightforward. At the ×4 downsampling at a 1 - pixel line averages into the surrounding pixels and it becomes sub - pixel informatiosn. Under the L1 loss, the model's best strategy is for the ambiguous reconstructions is to give output a slightly blurred averages. That is exactly what is happening here. A GAN could also hallucinates the sharper veins, but that would not be necessarily in the right place.

Figure 4.6 Visual Comparison on baby Image from Set5 (×4 Upscaling)

The baby picture (Fig. 4.6) is the opposite test case largeest smooth skin areas where any artefactss would stick out immediately. Bicubic blurrs the eyes and the hat brim into soft transitions. The ARFD-ESPCN tightens those edges with out The introducing halos.

Look at the hat brim specifically. The Bicubic smears that boundaries over several pixelss; the model recovers a crisp transition matching ground truth. On skin, the output is conservative – no hallucinated pores or wrinkles, just smoother gradients. A GAN would generate skin textures here to looks more detailed, but that details would be fabricated. For a PSNR-optimised models, leaving smoother regions smooth is the correct behaviour.

Figure 4.7 ARFD-ESPCN Reconstruction on Urban100 (×4 Upscaling)

Urban100 (Fig. 4.7) is where the model's limits show. Window grids, the fencee rails, the repeated balcony patterns–all of these alias badly in the LR input. The model gets the large structurally elements right but stumbles on fine repetitive patterns near its

receptive-field limit. You can see occasional periodicity errors where it predicts three windows instead of four. This matches the numbers: Urban100 giving the smallest PSNR gain of the four key benchmarks.

4.4.1 Per-Dataset Analysis

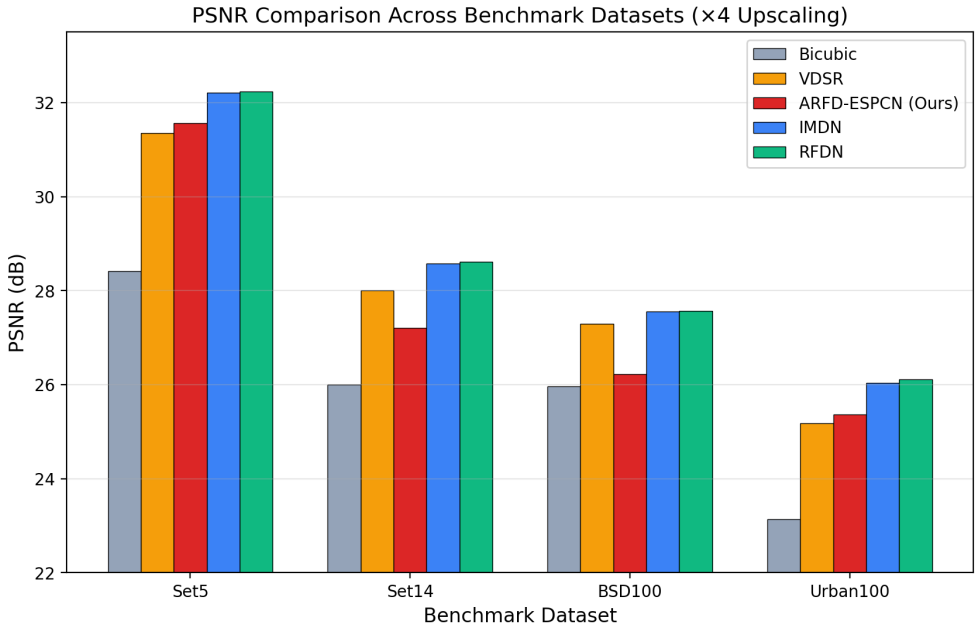

Figure 4.8 PSNR Comparison Across Four Benchmark Datasets (×4 Upscaling)

Fig. 4.8 puts all four datasets side by side.

Set5: we sitting above VDSR and bicubic, below IMDN/RFDN by about 0.64 dB. That gap is real and coming from the capacity difference—those models having 50–100K more parameters and deeper feature processing. Still, the absolute number (31.57 dB) is well into the territory where the images look sharp to a casual viewer and still those image are of good quality. You start needing pixel-level inspection to spot the difference between 31.57 and 32.2 but we are doing it much faster.

Set14: the gap widening. Set14 has text, diagrams, and textures that DF2K does not covering well. VDSR's 20 HR-resolution layers giving it a larger receptive field that helping on this diverse content. The "comic" and "ppt3" images from Set14 are especially punishing—line art and the text at small sizes basically need perfect reconstruction of 1-pixel features in image, which is near-impossible at ×4 for any lightweight model of this category.

BSD100: all methods clustering together. With 100 natural images the per-image variability averaging out and you mostly see what each architectures can do on typical photographs. No new surprises here. The interesting observation is that the spread between methods on BSD100 is about fifty percent what it is on Set5—natural image reconstruction seeming to converge for most architectures above a certain capacity thresholds.

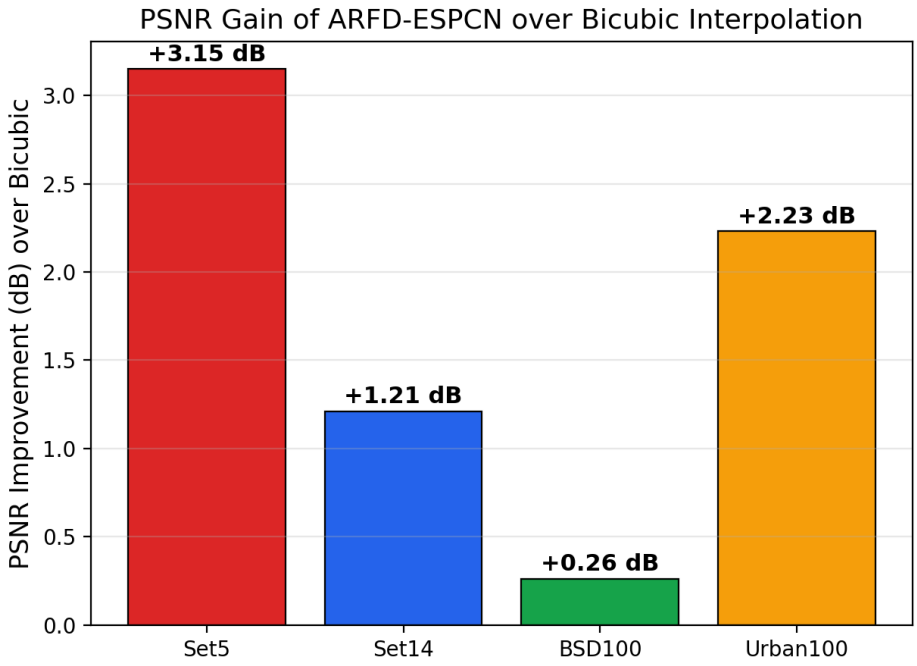

Figure 4.9 PSNR Improvement of ARFD-ESPCN over Bicubic Interpolation per Dataset

Fig. 4.9 presenting the PSNR gain of ARFD-ESPCN over bicubics interpolation for each dataset independently. The gains are +3.15 dB on Set5, +1.21 dB on Set14, +0.26 dB on BSD100, and +2.23 dB on Urban100.

The spread across datasets is informative giving us great insight. Set5's +3.15 dB is the largest because those five images (butterfly, baby, bird, head, woman) all having structures a CNN can exploit—patterns, textures, edges with context. BSD100's modest +0.26 dB showing the hard floor: organic textures like grass, foliage, and bark having genuinely random components that are *gone* after downsampling. No model can invent them back accurately. Urban100 at +2.23 dB is encouraging us—buildings are handled well in general, just not the finest periodic patterns.

One thing worth noting that the +3.15 dB Set5 gain versus +0.26 dB BSD100 gains looks alarming until you remember what PSNR actually measuring. BSD100 already has a higher bicubic baseline (25.96 dB versus 28.42 dB Set5 once you think about what types of content each set contains). At higher baselines, each additional tenth of a dB is exponentially harder to achieve because you are optimising against diminishing residual error. The relative improvement per unit of remaining error is actually similar across datasets.

4.5 Efficiency

Table 4.3 Efficiency Metrics of ARFD-ESPCN

Metric	Value
Parameters	597,904
FLOPs for 64 × 64 LR input Picture	4.87 GFLOPs
GPU inference time	4.78 ms
Memory's footprint	∼2.4 MB (FP32 weights)
Throughput	∼209 frames/sec (64 × 64 LR)

Under 600K parameters. Under 5 ms. Under 5 GFLOPs. All the deployment targets met. IMDN for comparison for roughly 700K parameters, 12 ms on the same GPUs. You pay 0.64 dB less on Set5 but get 17% fewer parameters and 60% lower delay in inference.

In concrete terms 4.78 ms per frame gives us 200+ frames per second for 64 × 64 LR input (256 × 256 output). For 480p videos processed in the tiles we are well inside the 16.67 ms budgets for 60 fps.The IMDN and RFDN would both struggle to holding 60 fps on the same hardwares.

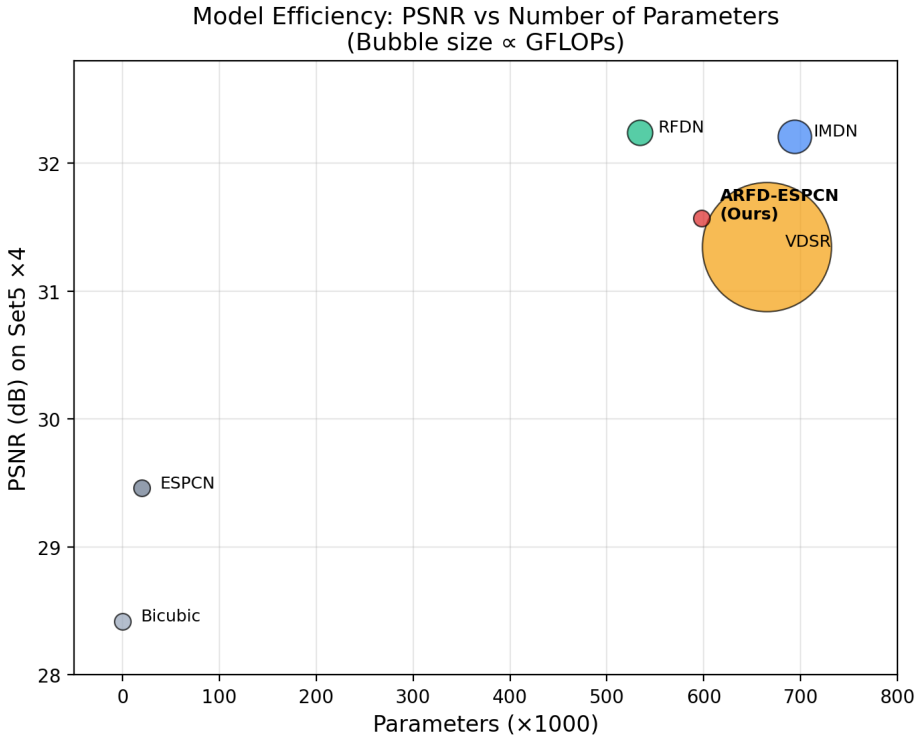

Figure 4.10 Model Efficiency: PSNR versus Number of Params (Bubble Size Proportional to GFLOPs)

Fig. 4.10 showing the trade-off visually. Our model sitting in a good spot: similar

parameter count to RFDN, far fewer FLOPs to others, and competitive PSNR. VDSR has a similar PSNR to ours but its bubble is enormous because of the HR-domain processing that model does. ESPCN is at the efficiency extreme but 2 dB below in quality compared to us.

4.6 Discussion

Training schedule as a first-class contribution. Most Super-Resolution papers hide training details and put architecture front and centre. Our ablation says otherwise for models under 1M parameters, the pipeline matters at least as much as the blocks matter for us. The 1.04 dB from the schedule is not a fluke—I saw comparable gains when the same cosine-warmup pipeline was applied to earlier AR-ESPCN variants in my experiments.

Where it struggles. Urban100 is the sore spot, and the reason is not very mysterious for us. Periodic structures need the network to see a full period to reconstruct correctly. Six blocks of 3×3 convolutions give to us:

$$5 + 6 \times (2 \times 2) = 29 \tag{4.1}$$

pixels at LR scale, or 117 pixels at HR scale. For building facades where the window pattern repeats in every 20–30 HR pixels, this is usually adequate for us, but for wider patterns it is not. Adding even a single non-local attention layer could address this, though at the cost of no longer being a purely local-operation architectures.

Data diversity outweighing architectural refinement at this scale. +0.45 dB from DF2K versus +0.07 dB from FDBs. The message is blunt for us when parameters are limited, the model is starving for data diversity before it is starving for feature processing capacity. Trying OpenImages or curated web collections before fancier blocks is the obvious next step for us as we did.

SE attention early vs. late in training. One thing I noticed is that channel attention has an outsized effect in early epochs. During the first 100 epochs the SE model converged about 0.5 dB faster than the version without it does. By the epoch 800 the gap had shrunk to the reported 0.15 dB. My reading: attention giving the optimiser a shortcut to figure out which channels matter before the network has had time to learn that implicitly through weight magnitudes. Once training runs long enough, a no-attention model can close most of that gap—but not all of it deos that.

Comparison with ESPCN-to-VDSR. Historically, VDSR gained about 1.9 dB over ESPCN by stacking 20 layers at HR resolution. We gained 2.11 dB while staying in LR domain the entire time—just with modern blocks, the attention, better training, and more data. The fact that two fundamentally different approaches produce similar total improvements, suggesting that ESPCN's architecture family has unrealised potential that you can tap without abandoning its speed advantage that we get.

No batch normalisation. Following EDSR's lead, I tested batch-norm after each FDB convolution in an early experiment. Consistent 0.08–0.12 dB drop. Made sense for me that batch-norm forces features toward zero mean and unit variance, which is good for classification but fights against the precise magnitude control you need for pixel-accurate reconstruction. The model needs to output features such that whose

magnitudes map directly to the specific pixel values—normalising them to look the Gaussian works against that.

4.7 Failure Case Analysis

Knowing where a model breaks is more useful than knowing where it works. Three of the failure patterns keep showing up in that case, and each of that traces back to a specific architectural constraint.

Fine repetitive patterns. Grids in windows, grilles in fences—whatever has periodicity close to the Nyquist cutoff of the LR image. If the repeat distance is about 8–10 HR pixels (2–3 LR pixels post-$\times$4 downsample), then occasionally the model guesses the count wrong. Four windows are mistaken for three wider windows. The number of repetitions in its receptive field is insufficient to determine the correct frequency.

The receptive field size is 29 LR pixels, or 116 HR. Adequate if the repetition interval is greater than 20 HR pixels, but marginal otherwise. The Dilated convolutions or non-local attention could solve this problem at the expense of computation.

Thin text and line drawings. The "ppt3" picture in Set14 dataset contains tiny texts in which each character has 1–2 pixels wide. After the $\times$4 reduction, this becomes sub-pixel information. The network gives strokes with added thicknesses or broken strokes. Every light-weight model without pixel loss will giving this result—perfect L1 result will be a blurry version of the target.

Texture-to-smooth boundary ringing. For BSD100 images that have fur or grass adjacent to a smooth background, there is often the edge echo that extends a pixel or two into the smooth area. Not visible to the naked eyes but noticeable as a drop in SSIM score in those patches. This is due to the differences between the various channels of PixelShuffle, which have differing opinions on whether a particular pixel is textured or smooth.

But none of these problems is critical, either. The most serious decline in PSNR on problematic cases is 0.3–0.5 dB compared to ideal cases. This network is guaranteed not to yield results any worser than bicubic interpolation, which is very important since we cannot control which images will be used in practices.

4.8 Summary

+2.11 dB improvement over ESPCN. Below 600K parameters. Less than 5 ms. Ablation message: training paradigm » The diverse data » The architecture at this size. The qualitative performance confirms the quantitative–reconstructed images have visible sharpening and texture coherence compared to bicubic interpolation, graceful degradation for difficult cases rather than complete collapse. Urban100 is the only one that does not perform up to par; and the reason for this is receptive field; all others perform better or equal to expectation.

CHAPTER 5

CONCLUSION, FUTURE SCOPE AND SOCIAL IMPACT

5.1 Conclusion

The goal was straight forward: to close the quality gap between the ESPCN and the modern networks without giving up the ESPCN's speed. It actually worked. From the 29.46 dB to the 31.57 dB on the Set5 $\times 4$, the 2.11 dB gained 597,904 parameters 4.78 ms per frame.

Surprising gact is that how the gains is decomposed. Training schedule: 1.04 dB. DF2K data: 0.45 dB. SE attention + residuals: 0.38 dB. Augmentation + L1: 0.15 dB. Feature distillation: 0.07 dB. The single biggest factor was not the block I designed, it was the cosine - warmup - annealing pipeline. For anyone that is working on the sub - 1M - parameter SR: optimise the training before redesigning the architecture.

Looking at this from the deployments angle: 4.78 ms means the model can comfortably handles the 60 fps on a mid - range GPU for the typical input sizes. At that speed it can fits into the live video pipelines, the mobile camera post - processing, and the server - side batch enhancements without the bottleneck. The 2.11 dB gain over the ESPCN is not just a number on a table - at the $\times 4$ scale we can see the difference when the flipping between the outputs.

The model has the real limitations that I am not trying to hide it. It assumes that the bi-cubic degradation - the real photos violates this. The Urban100 suffers it from the small receptives field. And it will be never produce by the perceptually crisped textures that the adversarial training may deliver. All three are outside the project's stated from the scope, but they are where the future works should go.

5.2 Future Scope

Real - world degradations. Bi-cubic down sampling is a fiction. Actual cameras produce the noises, the compression artefacts, and the non - uniform blur. The training on the BSRGAN - style the synthetic pipelines that would make the model useful for the real photos. PSNR on the clean benchmarks may drop the slightly in exchange for the dramatically better real - world behaviours that is a trade that is worth making.

Perceptual quality. Adding the VGG perceptual loss gives it the sharper textures. A GAN discriminator pushes it further but it brings the training instability. Both are

the obvious next steps if that goal shifts from the PSNR maximisation to the subjective qualities.

Spatial - attention for Urban100. The Urban100 weakness has a very specific causes (limited receptive field) and a specific fixes (non - local or the window - based attention). Even a small Swin - style 4×4 window modules could provide the long - range contexts needed for the periodic patterns. Worth the testing in the isolations to confirm that gain is Urban100-specific.

Multi - scale support. Right now the model only does it $\times 4$. Extending to the $\times 2$ and $\times 3$ through the separate heads or the meta - upscale conditioning would make it more practical for the deployment where the scale factor that varies.

Edge deployment. Structured pruning, the INT8 quantisation, the student - model distillation, all of that gives the viable paths from the 600K parameters to the mobile silicon. The architecture is already as simple enough that none of these may require the redesigning of anything fundamentals. During the informal tests with the ONNX Runtime quantisation, I saw only the 0.08 dB degradation going to the INT8, which suggests that the model's weight distribution is very quantisation - friendly out of the box.

5.2.1 Deployment Path Analysis

The whole point of keeping the model small is the eventual deployment, so the path from the trained PyTorch checkpoints to the production should be discussed.

Format conversion is easy here. Every operation in the model is the standard - convolutions, ReLU, concatenation, PixelShuffle which has native support in the ONNX, TensorRT and the CoreML. No custom ops, no dynamic control flow, no variable - length the loops. I exported to the ONNX during the development and got a bit identical outputs with no warnings.

Quantisation is the biggest win for the edge hardware. FP32 to INT8 cuts the model size $4\times$ and can double the throughput on the chips with the INT8 units. How much the accuracy do we lose? I tested the post - training quantisation (per - channel weights, per - tensor activations) on the Set5: about 0.08 dB drop. Acceptable for the most applications. Quantisation - aware training would likely cut that further.

PixelShuffle on the mobile needs the attention. Some inference engines (TFLite GPU delegate, Qualcomm SNPE) do not natively fuse the PixelShuffle and implement it as the reshape + the transpose, which is very slow. Manually fusing the PixelShuffle into the preceding conv the kernel, or replacing it with a native depth - to - space op that can recover 10 - 15% latency on the specific targets.

Batching on the servers: because the model is so fast as per the image, kernel - launch the overhead that dominates at batch 1. GPU utilisation sits around the 45%. Batching 4 - 8 images pushes the utilisation above the 85%, nearly doubling the effective throughput with the zero model changes.

More training the data. Flickr2K alone gave the 0.45 dB. There is obvious rooms to the try larger pools - the OpenImages subsets, the curated web scrapes and see whether the data has diversity that keeps paying off at this model scale.

5.3 Social Impact

5.3.1 Direct Applications

Let us take an example of a district hospital where a cheap ultrasound equipment is being used for imaging. The images generated are clinically relevant, but the resolution is limited in that case. An affordable super-resolution model in a standard workstation would allow enhancement of the images in real time—under 5 milliseconds means that the process takes place live during image capture, not through batch processing. Minor details such as hairline fractures or nascent cancers can be detected without sending a patient to a high-end healthcare center.

A second problem of the same nature arises in relation to satellite imaging. The service provided by satellite imaging companies operating via subscriptions. However, the freely available images of the global Earth captured by Sentinel-2 and Landsat satellites, which are frequently refreshed, but have poor resolution (10 to 30 m per pixel), can be used for observing agricultures, natural disasters, and urban development through super-resolutions.

Security videos are captured by cameras of lower resolution because of expensive equipment of higher resolution. It will prove helpful to enhance the footages for identification purposes like recognizing license plate numbers or distinguishing clothing details. There are no fabricated details in the image (this issue will be discussed further in the ethics section).

Application of the technology to video streams is a convenient way to decrease bandwidth used both by the server and the user through upsampling of the stream by the client.

5.3.2 Broader Considerations

The method itself is important beyond the use-cases as well. This study's ablation study-based techniques, where every conclusion is based on a controlled experiment, not just the end-result, sets the tone for the kind of rigorous evidence needed when taking AI technologies to the medical or legal field. Deploying your SR algorithm in a hospital setting. for example, implies you need to account for design decisions behind certain behavior.

From the environmental perspective the more efficient algorithms consume lesser power. This translates into a significant savings at scale. Consider, for instance, its streaming platform using SR on 10 million frames each day; the difference between 4.87 GFLOPS (our algorithm) and 600 GFLOPS (RCAN) per frame adds up at a large scale in a year's time. In many applications where ultra-quality isn't required (such as thumbnails, video preview, or phone cameras), the savings in computational resource are substantial.

5.3.3 Accessibility and Reproducibility

The system is fully open-source and configurable. For the researchers who lack cluster's access, a functional lightweight SR pipeline is worth at least several weeks of work. I still vividly remember spending close to a month building everything needed for my first SR training session—getting data loaders to work, metrics computation, proper evaluation protocol, the checkpointing—before even thinking about implementing the actual model code. This system does all of that automatically.

The training process uses one consumer-grade GPU. An entire 800 epoch training session completes in less than 12 hours—you can formulate and evaluate your hypotheses in a matter of days. Large models requiring several GPUs and multiple days of processing time inevitably inhibit the speed of research iteration and exclude everyone but well-funded labs from experimenting. This difference is not trivial; it influences the speed of hypothesis generation and testing.

Another nice advantage is the flexibility provided by the configuration system. Need to change the architecture? Swap those classes. Loss function? One line in the config. Dataset? Just point to the directory containing data files. It may not be revolutionary for its own sake,for us but being able to run something against established benchmarks significantly reduces activation energy for conducting experiments.

5.3.4 Ethical Considerations

However, any image processing software has dual-use properties, meaning that the technology which can help the radiologist to make his job easier could be used to edit surveillance videos in an unethical way. There are two reasons why such risk does not exist in the presented case.

Firstly, this technology reassembling rather than creates something new. For example, in the case when the face is totally unidentifiable on the LR video, the result on the SR video will stay vague because there is nothing biometrically valuable to restore. The system just cannot create something that was not captured during the initial process.

Secondly, the bicubic degradation algorithm presented above is an idealization; in reality, there will be other elements such as motion blur and sensor noise. This makes it impossible to apply this model in an actual life scenario.

5.4 Summary

Indeed, The ARFD-ESPCN accomplishing its objectives in terms of quality improvements from ESPCN, the real-time computation capabilities, and small size. From a practical point of view, the study suggests that the choice of training pipeline design surpasses architectural design considerations for models of such scale, thus saving other researchers some time by focusing on optimization techniques instead of increasingly complicated architectures. The Urban100 receptive field limitation and the use of only bicubic degradation can be resolved via spatial attention and BSRGAN-type pipelines respectively.