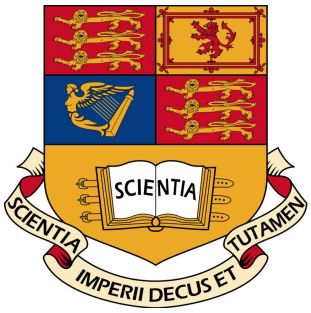


Imperial College London

Master of Research in Neurotechnology

Zifan Ning
Department of Bioengineering

BIOFLEXNET: Learnable Path Switching between Selectivity and Invariance Inspired by Synaptic Plasticity

Andriy Kozlov, Ph.D
Thesis Adviser
Department of Bioengineering
Imperial College London

Claudia Clopath, Ph.D
Thesis Co-adviser
Department of Bioengineering
Imperial College London

Date of Submission:

September 2025

Word Count: 5899 words.

Abstract

BIOFLEXNET: Learnable Path Switching between Selectivity and Invariance Inspired by Synaptic Plasticity

Zifan Ning
Imperial College London
September 2025

Advisers:
Andriy Kozlov, Ph.D
Claudia Clopath, Ph.D

Biological systems flexibly balance information preservation and compression, whereas standard convolutional networks enforce fixed convolution–pooling pipelines. This thesis introduces PlasticFlex, a framework that replaces every convolutional layer with a learnable routing module, enabling per-location gating between convolution and pooling within the same backbone. The design unifies artificial and spiking neural networks under a shared architecture, supporting both surrogate and non-surrogate training schemes without structural changes.

Experiments on MNIST, CIFAR-10, and ImageNet-100 show that PlasticFlex improves accuracy, robustness to adversarial perturbations, and training stability compared to standard baselines. Analyses of loss landscapes, Hessian spectra, and spiking dynamics further reveal that adaptive routing guides networks toward flatter minima, smoother curvature, and structured sparsity. These results suggest that flexible signal routing provides a biologically inspired mechanism for enhancing both the robustness and interpretability of deep networks.

Acknowledgments

I would like to thank my primary supervisor, Professor Andriy Kozlov, for giving me the opportunity to study at Imperial College London and for his guidance throughout the past year. His advice has been invaluable for shaping this work. I am also grateful to my second supervisor, Professor Claudia Clopath, for her supervision and feedback during the project.

I would also like to thank Professor Simon Schultz and Professor Guang Yang for their valuable discussions during my mid-term poster presentation, which provided many helpful suggestions for improving this work.

This work used the Imperial College Research Computing Service, DOI: 10.14469/hpc/2232.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background & Motivation

Biological sensory systems face a recurring trade-off: they need to capture fine local details while also staying stable when inputs vary.[5, 12] In vision, early cortical neurons are tuned to precise spatial features, whereas downstream neurons often generalize across shifts in position or scale.[6, 2] This balance between selectivity and invariance is not fixed but adapts depending on the context, task, and statistics of the input.[25]

Convolutional neural network (CNN) borrows this hierarchical idea but implement it rigidly.[11] Standard architectures alternate between convolution and pooling layers.[3] Pooling discards detail to gain invariance, [1] while convolution keeps local precision.[31] Once trained, the information flow in a CNN becomes fixed: data always passes through the same sequence of operations regardless of input content. This is efficient but ignores the possibility of adapting the information pathway itself.[9]

From a neuroscience and information-theoretic perspective, this rigidity is a simplification. [8] Real circuits route signals dynamically, reallocating bandwidth to regions of interest while allowing other regions to undergo more abstract processing.[22] In effect, the brain does not treat information flow as a single predetermined channel but as something that can be reorganized depending on the stimulus. Introducing this property into artificial networks could improve robustness and also provide a more realistic platform for studying biologically inspired computation.[18, 29]

1.2 Problem Statement

Despite progress in adaptive mechanisms such as attention [28] and dynamic convolution [10], most CNNs still apply the same operation to all spatial locations within a layer.[21] This global uniformity prevents the network from balancing selectivity and invariance in a spatially adaptive way. From an information-theoretic angle, the model commits to a single path of transformation, with no capacity to alter routing once the weights are fixed. Existing routing methods rarely quantify how such rigidity impacts robustness,

interpretability, or formal verifiability.[35, 36] In spiking neural network (SNN), where computation unfolds over time, the lack of routing flexibility is even more restrictive, limiting both efficiency and the diversity of information flow.[37]

1.3 Research Objective

This work introduces PlasticFlex, a routing module that lets each spatial location decide between convolution and pooling during downsampling. In contrast to a single fixed channel of transformation, PlasticFlex allows multiple candidate pathways and learns where to allocate information flow. The design runs in both ANN and SNN settings without changing the backbone, enabling controlled comparisons. The objectives are: (i) test whether adaptive routing improves accuracy and robustness across datasets; (ii) measure interpretability through mask sensitivity analysis; (iii) derive theoretical bounds showing routing does not break certifiability; (iv) examine how spiking activations interact with routing to affect sparsity and efficiency.

1.4 Contributions

The main contributions are:

(i) PlasticFlex Layer: A biologically inspired information-routing module that adaptively selects between convolutional and pooling transformations at each spatial location, acts as a local router that controls whether information is preserved with high fidelity or compressed for invariance, allowing the effective information channel to vary across space.

(ii) Routing Configurability: Masks can be learned or fixed, enabling direct comparisons of fixed versus adaptive information flow.

(iii) Unified ANN–SNN Backbone: A shared architecture that supports both spiking and non-spiking activations, ensuring fair evaluation across operational modes.

(iv) Mask Sensitivity Metric: Closed-form bounds on outputs ensure compatibility with verification tools such as IBP and CROWN, showing that flexibility in routing does not undermine certifiability.

(v) Comprehensive Evaluation: Systematic ablation across datasets, depths, and activation types, isolating the effect of allowing information flow to remain flexible rather than static.

Chapter 2

Related Works

2.1 Selectivity–Invariance in Vision Systems

Selectivity denotes strong responses to specific features, while invariance ensures stable recognition under transformations.[6] Both are crucial: selectivity enables discrimination, invariance supports generalisation.[3]

Classical CNNs fix this trade-off by combining convolution and pooling. Convolutions match local patterns, pooling removes detail to build invariance. Once trained, the same balance is applied everywhere, regardless of local input variability.[32, 4]

In biology, this balance emerges gradually along the ventral stream: V1 neurons are selective to orientations,[24] whereas IT neurons show invariance to position and scale. [6] Crucially, these properties adapt with task and context, suggesting that artificial models should also allow routing decisions to change dynamically.

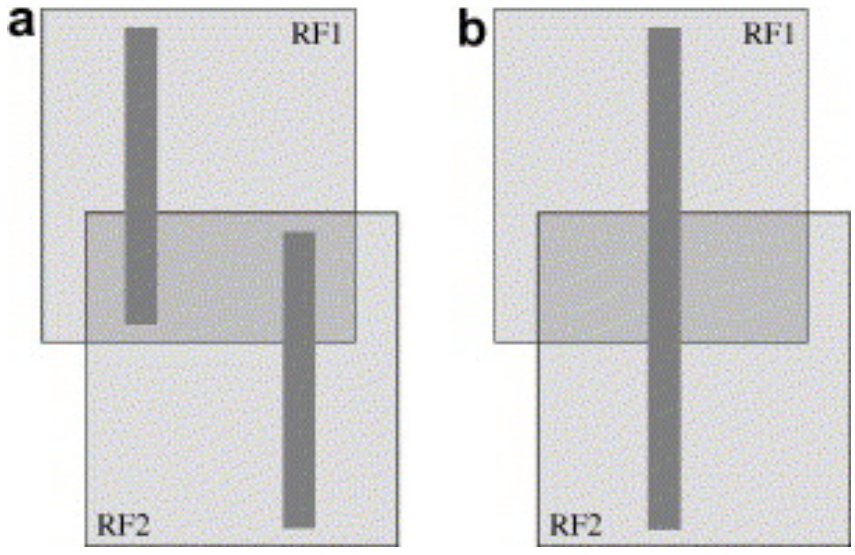


Figure 2.1 Invariance vs. selectivity.[14] Imagine two cells N1 and N2, with receptive fields RF1 and RF2, which signal vertical bars with invariance to horizontal position ("phase"). The situations in panels (a) and (b) are indistinguishable, given only the individual activity levels of N1 and N2.

2.2 Dynamic Routing Mechanisms

Dynamic routing [10] alters the computational path per input or region, instead of applying a uniform operation. Early methods include conditional computation, where filters are activated by gating signals.[7] Capsule networks added routing-by-agreement. Later work explored attention and learnable masks [30] for choosing kernels or resolutions.

Biological systems also reroute signals depending on context [23] and attention [27], shifting receptive fields or connectivity to prioritise detail or invariance. [25] This motivates artificial routing mechanisms that can adaptively tune information flow.[15]

Challenges remain: routing must be trainable, efficient, and stable. Existing solutions (soft gating [13], straight-through estimators [34], reinforcement updates [16]) improve adaptiveness but do not directly quantify how routing shapes the selectivity–invariance balance.

2.3 SNN and Biological Plausibility

SNNs transmit information through spikes, introducing temporal coding and natural sparsity.[33] This matches biological computation and motivates their use in energy-efficient hardware.[20]

Neurons in SNNs integrate inputs until a threshold triggers a spike, providing an event-driven substrate closer to cortical dynamics.[12] Such models enable study of temporal processing and plasticity mechanisms.[29]

Training is difficult due to the non-differentiable spike function.[26] Surrogate gradients or ANN-to-SNN conversion partly address this,[38, 17] but often obscure the link between spiking activity and functional roles such as selectivity or invariance.[9]

Combining SNNs with dynamic routing opens a path to test how spiking and routing jointly shape detail preservation, robustness, and efficiency.[19] Our work builds directly on this connection, embedding spiking activations into a flexible routing framework under a unified ANN–SNN architecture.

Chapter 3

Methodology

3.1 Overview of the Approach

We extend standard convolutional networks by replacing every layers with PlasticFlex, a routing module that offers two parallel transformations: a convolution branch and a pooling branch. At each spatial location, a learnable mask decides how much information flows through each branch. Unlike classical CNNs, where information follows a fixed path once trained, this design allows the information flow to remain adaptive across all layers, as shown in Figure 3.1.

The framework is paired with a configurable activation unit that can operate in either ANN (ReLU) or SNN (spiking) mode. Two independent switches therefore control: (i) routing behaviour, by choosing whether the mask remains trainable or is frozen, and (ii) activation dynamics, by selecting spiking or non-spiking functions. This separation enables systematic analysis of routing flexibility and spiking effects under the same backbone.

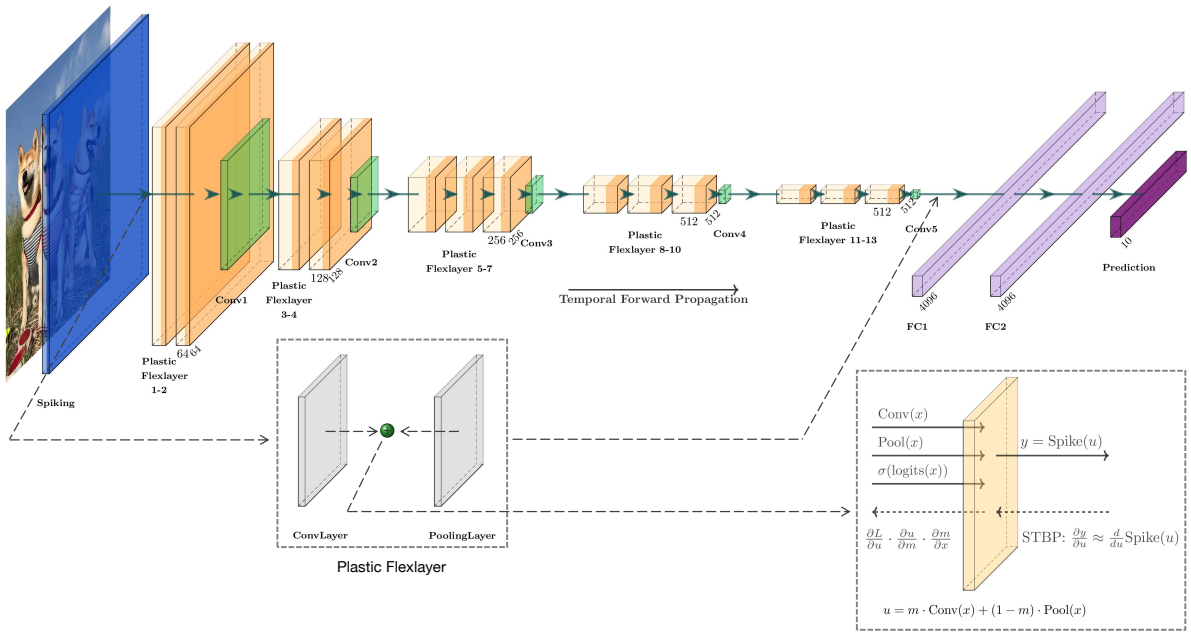


Figure 3.1 Framework of BioFlexNet.

3.2 Network Architecture

The backbone follows VGG-style designs in three depth variants: VGG6, VGG11, and VGG16. Each convolutional block is implemented using PlasticFlex instead of a fixed convolution or max-pooling. The PlasticFlex module computes a standard convolution and a max-pooling operation in parallel, then combines their outputs via the spatial mask derived from local features. Downsampling is handled naturally by stride-2 operations within PlasticFlex when resolution reduction is required.

After each PlasticFlex layer, batch normalization is applied, followed by the activation specified by the operational mode (ReLU for ANN, spiking activation for SNN). Fully connected layers are placed after the convolutional blocks, projecting to the dataset's class space. By holding the backbone structure constant, observed performance differences can be traced to the routing and activation mechanisms rather than to network capacity. For complete model architectures and hyper-parameters, please see Appendix B

3.3 Flex Routing Mechanism

Each layer is implemented as a PlasticFlex module with two parallel branches: a convolution branch $F_{\text{conv}}(\cdot)$ and a pooling branch $F_{\text{pool}}(\cdot)$. A spatial routing mask $M \in [0,1]^{H\times W}$ decides, for every location (i,j), how much information flows through each branch. The mask is obtained from learnable logits Z derived from the input features, followed by an activation function $\sigma_{\text{mask}}(\cdot)$ such as sigmoid or hard-sigmoid:

$$M = \sigma_{\text{mask}}(Z), \quad Z = f_{\text{mask}}(X), \tag{3.1}$$

where X is the input, and $f_{\text{mask}}(\cdot)$ is a learnable mapping (e.g., 1×1 convolution or shared parameters).

The output Y is a convex combination of the two branches:

$$Y = M \odot F_{\text{conv}}(X) + (1 - M) \odot F_{\text{pool}}(X), \tag{3.2}$$

with $\odot$ denoting element-wise multiplication. This formulation allows different spatial positions within the same feature map to follow different routes, making feature extraction adaptive rather than fixed.

To analyse routing behaviour, two statistics are monitored during training and evaluation:

- **Convolution ratio:** proportion of positions routed to the convolution branch,

$$r_{\text{conv}} = \frac{1}{HW} \sum_{i=1}^{H} \sum_{j=1}^{W} \mathbb{1}\{M_{i,j} > 0.5\}. \tag{3.3}$$

- **Binariness:** degree to which mask values approach $\{0, 1\}$,

$$b_{\text{mask}} = 1 - \frac{4}{HW} \sum_{i=1}^{H} \sum_{j=1}^{W} M_{i,j}(1 - M_{i,j}). \tag{3.4}$$

These metrics provide a direct view of how the network allocates detail-preserving versus invariant processing, serving as the basis for later interpretability analysis.

Algorithm 1: PlasticFlex Forward

Input: Input feature map X

Output: Output feature map Y

$C \leftarrow F_{\text{conv}}(X)$ `// convolution branch (stride 1 or 2)`

$P \leftarrow F_{\text{pool}}(X)$ `// pooling branch (aligned in channels)`

if *mask is* ***frozen*** **then**

  $M \leftarrow M_{\text{frozen}}$ `// stored binary mask`

else

  $Z \leftarrow f_{\text{mask}}(X)$ `// logits from features`

  $M \leftarrow \sigma_{\text{mask}}(Z)$ `// sigmoid / hard-sigmoid / STE`

$Y \leftarrow M \odot C + (1 - M) \odot P$

return Y

3.4 Mask Behaviour Control

Each PlasticFlex module produces its routing mask from learnable logits, transformed by an activation function $\sigma_{\text{mask}}(\cdot)$. Two activation types are supported:

- **Sigmoid:** a continuous mapping in $(0, 1)$, allowing smooth gradients for stable optimisation.
- **Hard-sigmoid:** a piecewise-linear approximation that saturates near 0 and 1, encouraging more discrete routing behaviour during training.

The learnability of the mask can be configured in two modes:

- **Trainable mask:** the mask parameters remain learnable throughout training, enabling dynamic adaptation of routing at all epochs.
- **Frozen mask:** the mask is trained for an initial number of epochs, after which its parameters are fixed, enables testing whether ongoing adaptation improves performance or stability.

To prevent mask collapse—a degenerate case where all positions route through only one branch—a regularisation term is added to the loss:

$$\mathcal{L}_{\text{mask}} = \lambda_{\text{mask}} \cdot \frac{1}{HW} \sum_{i=1}^{H} \sum_{j=1}^{W} (M_{i,j} - \tau)^2, \tag{3.5}$$

where λ_{mask} controls the regularisation strength, and τ is a target routing ratio (e.g., 0.5 for balanced routing). This term penalises extreme bias towards one branch while preserving flexibility to adapt to the task.

During analysis, the evolution of the mask is monitored through:

- **Convolution ratio trajectory:** r_{conv} over epochs, indicating routing preference changes.
- **Binariness trajectory:** b_{mask} over epochs, reflecting the discreteness of routing decisions.

Tracking r_{conv} and b_{mask} over epochs reveals how routing preferences evolve and whether the mask stabilises under different control strategies.

3.5 Spiking Behaviour

The framework supports both ANN and SNN modes by selecting the activation function after each layer: ReLU for ANN, SpikingActivation for SNN. In ANN mode, a standard ReLU is used. In SNN mode, the activation is replaced by a SpikingActivation unit that introduces membrane potential dynamics and threshold-based firing.

We implement two operation modes:

- **Surrogate gradient mode:** At each time step t, the membrane potential U_t is compared against a threshold θ (set to 1.0 in all experiments). A spike is generated when the threshold is crossed:

$$S_t = \mathbb{1}\{U_t > \theta\}. \tag{3.6}$$

  Because the derivative of this step function is zero almost everywhere, we use a surrogate gradient during training:

$$\frac{\partial S_t}{\partial U_t} \approx \alpha \cdot \tanh(U_t), \tag{3.7}$$

  where α controls the slope, preserves binary spike dynamics in the forward pass while enabling gradient-based training.
- **Continuous mode:** Instead of hard thresholding, the output is bounded through a clamping function:

$$S_t = \text{clip}(U_t, 0, 1). \tag{3.8}$$

  This yields a continuous analogue of spiking behaviour, removes the need for surrogate gradients but loses the explicit event-driven character of spiking.

In both modes, the spiking activation is applied channel-wise at each spatial location. To evaluate temporal behaviour, the network is unrolled over T discrete steps, and outputs are averaged:

$$Y = \frac{1}{T} \sum_{t=1}^{T} f_{\text{SNN}}(X_t). \tag{3.9}$$

This design enables direct ANN–SNN comparison under the same backbone and highlights the trade-off between biological fidelity and training stability.

Algorithm 2: SNN Temporal Unrolling (if $T > 1$)

Input: Input x, steps T

Output: Prediction y

$y_{\text{acc}} \leftarrow 0$ **for** $t = 1, \ldots, T$ **do**

  $x_t \leftarrow \text{Encode}(x, t)$ `// rate/latency/identity`

  $h_t \leftarrow \text{Backbone} + \text{PlasticFlex}(x_t)$ $s_t \leftarrow \text{SpikingActivation}(h_t)$ `// STBP or non-STBP`

  $y_{\text{acc}} \leftarrow y_{\text{acc}} + \text{Head}(s_t)$

$y \leftarrow \text{Aggregate}(y_{\text{acc}})$ `// mean or sum`

return y

3.6 Training Procedure

All models are trained using stochastic gradient descent with momentum (SGD+Momentum), a momentum coefficient of 0.9, and weight decay of 5×10^{-4}. The initial learning rate η_0 is set to 0.1 for CIFAR-10 and ImageNet100, and 0.01 for MNIST. A cosine annealing schedule gradually reduces the learning rate to zero over the course of training:

$$\eta_t = \eta_{\min} + \frac{1}{2}(\eta_0 - \eta_{\min}) \left[1 + \cos\left(\frac{\pi t}{T} \right) \right], \tag{3.10}$$

where t is the current epoch and T is the total number of epochs.

Data preprocessing follows standard protocols for each dataset:

- **CIFAR-10 / ImageNet100:** random cropping with 4-pixel padding, random horizontal flipping, and per-channel normalization.
- **MNIST:** zero-padding to 32×32 resolution for compatibility with VGG backbones, followed by normalization to zero mean and unit variance.

All models are trained with cross-entropy loss:

$$\mathcal{L}_{\text{CE}} = -\frac{1}{N} \sum_{n=1}^{N} \sum_{c=1}^{C} y_{n,c} \log p_{n,c}, \tag{3.11}$$

where $y_{n,c}$ is the ground-truth label and $p_{n,c}$ is the predicted probability for class c.

For SNN variants trained with surrogate gradients (SG), the backpropagation step applies the surrogate derivative defined in Eq. (6), replacing the zero derivative of the spike function. Non-surrogate SNNs and ANN variants use standard backpropagation without modification.

To enable interpretability analysis, mask statistics (convolution ratio and binariness) are recorded at the end of each epoch for all PlasticFlex modules. For robustness evaluation, adversarial attacks are applied post-training to the final model checkpoints.

Algorithm 3: Training Step with Optional Mask Freezing

Input: Mini-batch (x, y), epoch e

Output: Updated parameters

$\hat{y} \leftarrow \text{Model}(x)$ `// forward with PlasticFlex + activation`

$\mathcal{L} \leftarrow \mathcal{L}_{\text{CE}}(\hat{y}, y)$ **if** *mask regularizer is enabled* **then**

  $\mathcal{L} \leftarrow \mathcal{L} + \lambda_{\text{mask}} \cdot \mathcal{L}_{\text{mask}}(M)$

Backprop($\nabla\mathcal{L}$) ; OptimizerStep()

if $e = E_f$ *and mask is learnable* **then**

  $M_{\text{frozen}} \leftarrow \mathbb{1}\{M \geq 0.5\}$ `// binarize and store`

  freeze mask parameters and skip logits branch henceforth

3.7 Theoretical Analysis

The aim of this section is to examine whether the flexibility introduced by the PlasticFlex layer changes its fundamental safety properties. In particular, we focus on deriving bounds and sensitivity measures that can be formally verified. The complete derivations are provided in Appendix A

3.7.1 Certifiability of PlasticFlex Layer

One concern with a flexible routing layer is whether its dynamic structure could break the guarantees needed for safe or verifiable deployment. In other words, even if the mask changes with the input, can we still say with certainty that the output will remain within a known range? This section shows that we can: for any bounded input, the PlasticFlex output can be enclosed in a closed-form bound that does not depend on the exact mask values.

The output at spatial position i is:

$$y_i = m_i \cdot C_i + (1 - m_i) \cdot P_i \tag{3.12}$$

where $m_i \in [0, 1]$ is the routing mask, and C_i, P_i are the convolution and pooling outputs.

If each component lies in a known interval:

$$C_i \in [l_c, u_c], \quad P_i \in [l_p, u_p], \quad m_i \in [l_m, u_m] \subseteq [0, 1] \tag{3.13}$$

then the output is guaranteed to be within:

$$y_i \in [\min(\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4),\ \max(\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4)] \tag{3.14}$$

where the four possible extremes come from the combinations:

$$\begin{aligned}
\mathcal{E}_1 &= l_m \cdot l_c + (1 - l_m) \cdot l_p \\
\mathcal{E}_2 &= u_m \cdot l_c + (1 - u_m) \cdot l_p \\
\mathcal{E}_3 &= l_m \cdot u_c + (1 - l_m) \cdot u_p \\
\mathcal{E}_4 &= u_m \cdot u_c + (1 - u_m) \cdot u_p
\end{aligned} \tag{3.15}$$

This bound holds no matter how the mask changes during inference. It means the PlasticFlex layer remains compatible with interval-based certification tools such as IBP or CROWN, ensuring that its structural flexibility does not undermine formal robustness guarantees.

3.7.2 Interval Bound Propagation

Having established that the output is certifiably bounded, we next look at how these bounds behave under input perturbations. The question here is whether we can track the range of possible outputs when the inputs are only approximately known. This forms the link between theoretical guarantees and robustness certification.

Recall that the layer output is given by:

$$y_i = m_i \cdot C_i + (1 - m_i) \cdot P_i \tag{3.16}$$

where m_i is the routing mask and C_i, P_i are the outputs from the convolution and pooling branches.

Assume that the bounds for each term are known from the previous bound-propagation step:

$$C_i \in [l_c, u_c], \quad P_i \in [l_p, u_p], \quad m_i \in [l_m, u_m] \subseteq [0, 1] \tag{3.17}$$

Given these intervals, the tightest output bound can be obtained by evaluating all combinations of mask and branch extremes:

$$y_i \in [\min(\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4),\ \max(\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4)] \tag{3.18}$$

where:

$$\begin{aligned}
\mathcal{E}_1 &= l_m \cdot l_c + (1 - l_m) \cdot l_p \\
\mathcal{E}_2 &= u_m \cdot l_c + (1 - u_m) \cdot l_p \\
\mathcal{E}_3 &= l_m \cdot u_c + (1 - l_m) \cdot u_p \\
\mathcal{E}_4 &= u_m \cdot u_c + (1 - u_m) \cdot u_p
\end{aligned} \tag{3.19}$$

This formulation allows PlasticFlex to be inserted into interval bound propagation frameworks without modification. Since the bound is computed in closed form, it can be applied repeatedly through the network, enabling full-network stability analysis under input perturbations.

3.7.3 Mask Sensitivity Analysis

While certifiability addresses safety, it does not tell us how much the routing mask actually matters to the network's function. Mask sensitivity analysis measures how strongly the output depends on each mask element, helping to interpret whether routing decisions are critical or redundant in different regions. To understand how strongly these routing decisions affect the output, we examine the sensitivity of y_i with respect to m_i.

The output of a PlasticFlex unit is:

$$y_i = m_i \cdot C_i + (1 - m_i) \cdot P_i \tag{3.20}$$

Taking the derivative with respect to m_i gives:

$$\frac{\partial y_i}{\partial m_i} = C_i - P_i \tag{3.21}$$

The magnitude of this derivative indicates the effect of a small change in the mask value on the output. We define the local *mask sensitivity score* as:

$$S_i := |C_i - P_i| \tag{3.22}$$

and compute the global mean over all positions:

$$\bar{S} := \frac{1}{N} \sum_{i=1}^{N} S_i \tag{3.23}$$

High S_i values indicate positions where the routing choice is critical for the output, while low values suggest that the two branches produce similar results. This measure can be used as an interpretability tool, and may also guide regularisation strategies for regions with high sensitivity.

Chapter 4

Experiments

4.1 Datasets

We conducted experiments on three datasets with increasing complexity to assess the performance, interpretability, and robustness of the proposed architecture.

4.1.1 MNIST

MNIST consists of 70,000 grayscale handwritten digit images (28×28), with 60,000 for training and 10,000 for testing. Despite its simplicity, it provides a clean setting to validate spiking activations and flexible routing under low-dimensional visual patterns.

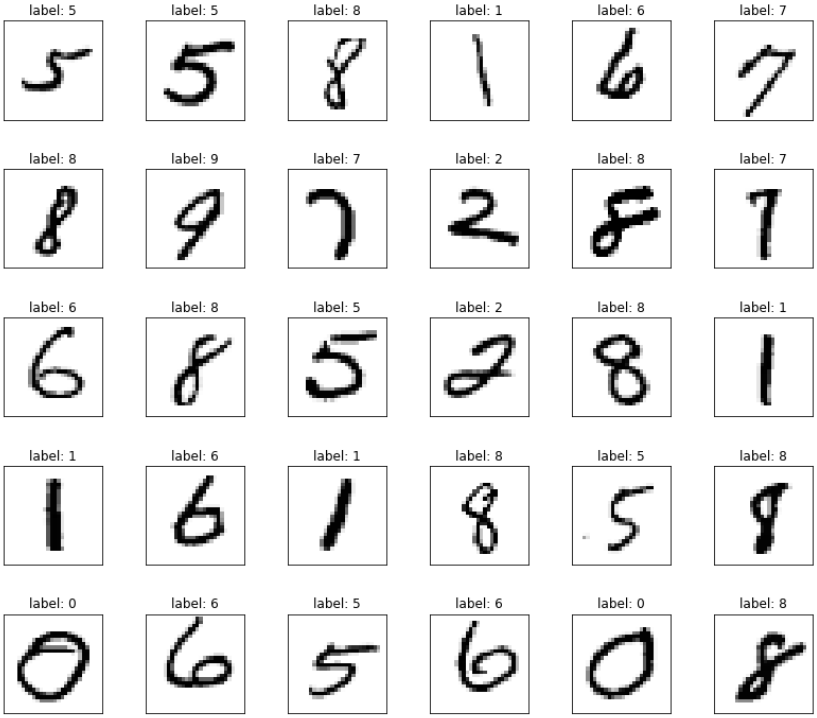


Figure 4.1 MNIST dataset snapshot.

4.1.2 CIFAR-10

CIFAR-10 contains 60,000 RGB images (32×32) evenly split across 10 categories, with 50,000 for training and 10,000 for testing. Compared to MNIST, it introduces colour and higher variability, providing a benchmark to test how Flex routing adapts under richer spatial statistics.

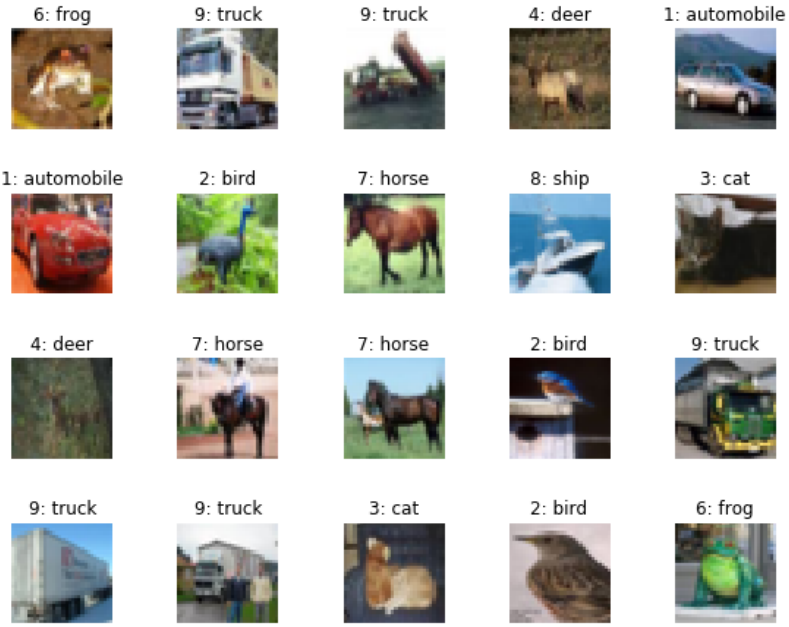


Figure 4.2 CIFAR-10 dataset snapshot.

4.1.3 ImageNet-100

ImageNet-100 is a balanced subset of 100 classes from ImageNet, with about 1,300 training and 50 validation images per class (224×224). It evaluates the scalability of Flex routing to higher-resolution inputs and a larger label space, and tests whether interpretability and robustness trends generalise beyond small-scale datasets.

4.2 Experimental Setup

We evaluate PlasticFlex under controlled settings by varying three main factors: (i) backbone depth (VGG6, VGG11, VGG16), (ii) routing flexibility (standard conv+pool vs. PlasticFlex with learnable or frozen masks), and (iii) activation mode (ANN with ReLU, SNN with surrogate gradients, or SNN with non-surrogate clamping). This design

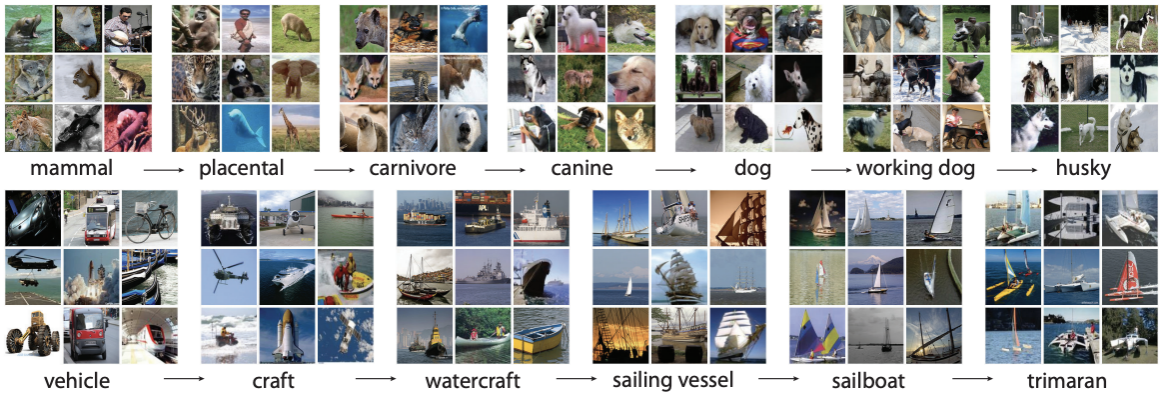


Figure 4.3 ImageNet dataset snapshot.

isolates the contributions of structural flexibility and spiking behaviour across different model capacities and dataset scales (MNIST, CIFAR-10, ImageNet-100).

4.2.1 Training Details

All models are trained with stochastic gradient descent (SGD) using an initial learning rate of 0.1, momentum 0.9, and weight decay 5×10^{-4}. The learning rate decays by 0.1 at scheduled epochs depending on dataset size. Training runs for 120 epochs on MNIST, 120 epochs on CIFAR-10, and 180 for ImageNet-100. Batch sizes are 128 (MNIST, CIFAR-10) and 256 (ImageNet-100). Weights are initialised with Kaiming normal initialisation.

For SNN variants, activations are unrolled over $T = 10$ timesteps with a spiking threshold $\theta = 1.0$. Surrogate training follows the surrogate gradient rule with slope $\alpha = 0.3 \tanh(U_t)$, while non-surrogate mode replaces binary spikes with a continuous clamping function. PlasticFlex masks are initialised uniformly and regularised with an ℓ_1 penalty to avoid collapse. All experiments are run on NVIDIA A100 GPUs with mixed precision.

4.2.2 Baselines and Variants

We use a factorised design so that each choice can be isolated while keeping depth and channel widths fixed.

1. Backbone depth. VGG6 / VGG11 / VGG16.

2. Routing (fixed vs. flexible).

    - Standard CNN: all layers are conventional convolutions (with pooling where needed).

- PlasticFlex: every convolutional layer is replaced by PlasticFlex.

3. Mask parameterisation and schedule.
   - Parameterisation: sigmoid; hard-sigmoid with STE; deterministic binary mask.
   - Schedule: trainable for all epochs; or frozen after a pre-defined epoch E_f (binarise at 0.5 and stop updating).
4. Activation mode (ANN vs. SNN).
   - ANN: ReLU.
   - SNN (STBP): binary spikes with surrogate gradients.
   - SNN (continuous): bounded continuous surrogate (no surrogate gradient).
5. Datasets. MNIST, CIFAR-10, ImageNet-100.

This factorisation allows main effects and interactions (routing, mask behaviour, neural dynamics) to be assessed under a consistent backbone.

4.3 Evaluation Protocols

We evaluate models along four dimensions:

1. Accuracy and balanced accuracy. The standard top-1 accuracy is defined as

$$\mathrm{Acc} = \frac{1}{N}\sum_{n=1}^{N} \mathbb{1}\{\hat{y}_n = y_n\}, \tag{4.1}$$

   where $\hat{y}_n$ is the predicted class and y_n is the ground truth. Balanced accuracy averages recall across classes to address class imbalance:

$$\mathrm{Acc}_{\mathrm{bal}} = \frac{1}{C}\sum_{c=1}^{C} \frac{TP_c}{TP_c + FN_c}, \tag{4.2}$$

   where TP_c and FN_c denote true positives and false negatives for class c.

2. Robustness. Adversarial robustness is measured under FGSM, PGD, and SPGD attacks. The FGSM perturbation is given by

$$x' = x + \epsilon \cdot \mathrm{sign}\left(\nabla_x \mathcal{L}(f_\theta(x), y)\right), \tag{4.3}$$

where ϵ is the perturbation budget. PGD applies multiple projected gradient steps:

$$x^{t+1} = \Pi_{\mathcal{B}(x,\epsilon)} \left(x^t + \alpha \cdot \text{sign} \left(\nabla_x \mathcal{L}(f_\theta(x^t), y) \right) \right), \tag{4.4}$$

where $\Pi_{\mathcal{B}(x,\epsilon)}$ projects onto the ϵ-ball around x. SPGD uses stochastic gradient estimates with random direction vectors u_k:

$$\nabla_x \mathcal{L}(f_\theta(x), y) \approx \frac{1}{K} \sum_{k=1}^{K} \frac{\mathcal{L}(f_\theta(x + \delta u_k), y) - \mathcal{L}(f_\theta(x - \delta u_k), y)}{2\delta} u_k, \tag{4.5}$$

which enables gradient-free adversarial optimisation.

3. Interpretability. We monitor routing statistics (convolution ratio, mask binariness), gradient flow, and Hessian spectra to assess how routing decisions and spiking dynamics influence learning.

4. Sparsity and efficiency. For SNNs, the average spike rate is recorded. For PlasticFlex, the degree of mask binarisation indicates the extent of computational sparsity, serving as a proxy for efficiency.

4.4 Computational Resources

All experiments were conducted on the Imperial College London High Performance Computing (HPC) facility (Research Computing Service, DOI: 10.14469/hpc/2232). The compute nodes used in this study were equipped with dual-socket AMD EPYC 7742 processors (64 cores per socket, 256 hardware threads in total) and 1.0 TiB of system memory, running Linux kernel 4.18. GPU-accelerated training was performed on NVIDIA A100 GPUs (40 GB memory). Jobs were submitted via the PBS scheduler with typical resource requests of 2 CPU cores, 8 GB RAM, and 1 GPU per job, with a maximum walltime of 32 hours.

The complete hardware and software resources are listed in Appendix D

Chapter 5

Results

5.1 Classification Performance

5.1.1 MNIST

Figure 5.1 shows the training and validation curves for all MNIST variants. Across architectures, all models reached 99.6% balanced accuracy, and 100.0% top-1 accuracy (Appendix C.1) on the validation set, indicating that neither the spiking activation nor the flexible routing impaired classification performance on this dataset.

A consistent effect was observed in the loss dynamics: models with the PlasticFlex mechanism showed more stable validation loss, even after training accuracy saturated. This suggests that adaptive routing between convolution and pooling paths provides a regularising effect, reducing oscillations during optimisation.

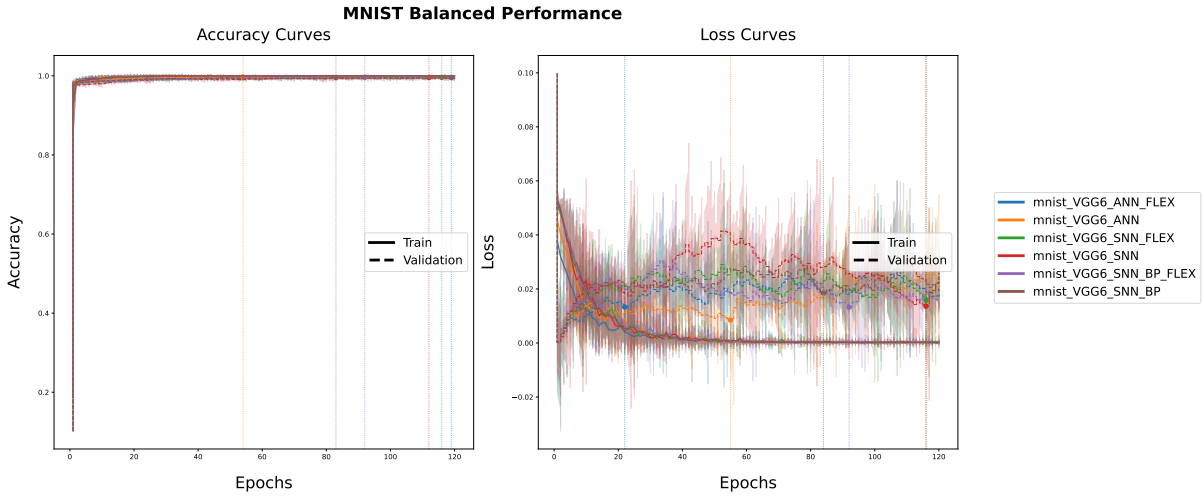


Figure 5.1 MNIST balanced training curves (VGG6 backbone). Left: balanced accuracy for training and validation. Right: balanced loss for training and validation.

5.1.2 CIFAR-10

Figure 5.2 shows the training dynamics on CIFAR-10 with VGG11 backbones. Among ANN models, the Flex-enabled variant reached the highest validation balanced accuracy (88.8%), slightly above the standard ANN baseline (87.9%). A similar trend appeared

in SNNs: the spiking network with Flex achieved 86.0%, compared to 84.7% for the best non-Flex spiking baseline (SNN-BP).

Across both ANN and SNN settings, the flexible routing mechanism consistently improved balanced accuracy. Although the gains were modest, the effect was reproducible, suggesting that spatially adaptive routing between convolution and pooling helps preserve discriminative information. The loss curves further show smoother convergence for Flex variants, indicating that Flex stabilises optimisation while maintaining accuracy.

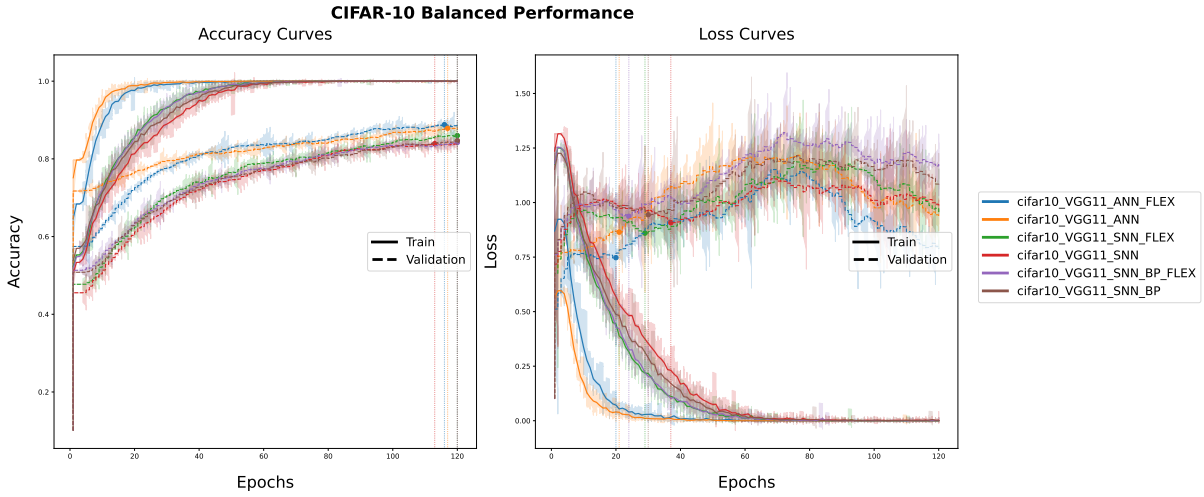


Figure 5.2 CIFAR-10 training curves (VGG11 backbone). Left: balanced accuracy for training and validation. Right: balanced loss for training and validation.

5.1.3 ImageNet-100

Figure 5.3 shows the training and validation curves on ImageNet-100. Validation accuracy saturates around 78.5%–80.6% for ANN variants and slightly lower for SNNs, showing that the models cannot fully capture the complexity of ImageNet-100. The gap between training and validation curves is larger than in MNIST and CIFAR-10, indicating weaker generalisation. Loss values converge but plateau at higher levels, consistent with the more difficult task.

Flex provides a modest but consistent benefit. ANN+Flex and SNN+Flex converge more smoothly and reach slightly higher validation accuracy than their baselines. SNN+BP with Flex also trains more stably than SNN+BP without Flex, suggesting that routing reduces instability. Performance on ImageNet-100 remains limited, but Flex improves optimisation and gives small accuracy gains.

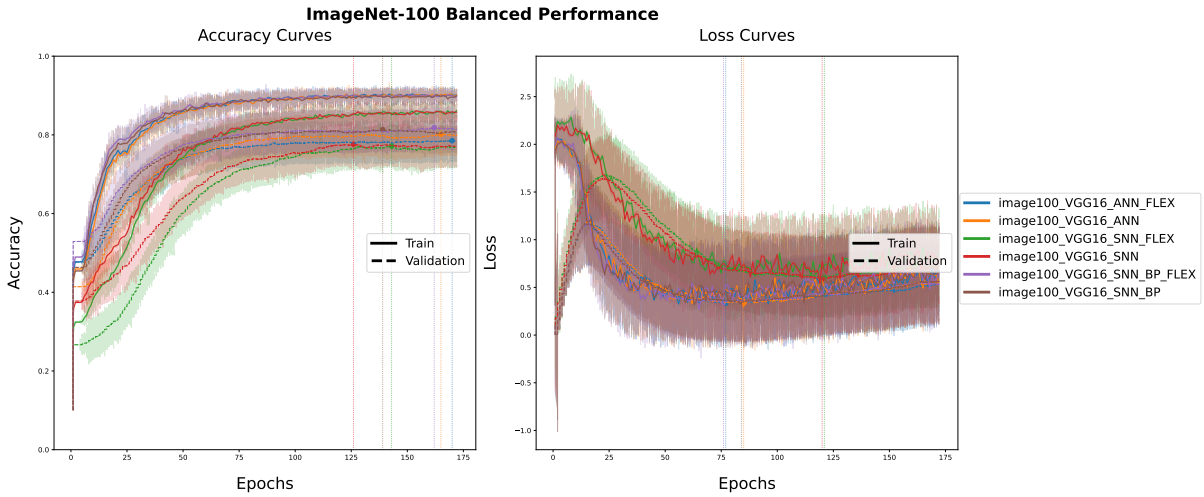


Figure 5.3 ImageNet-100 training curves (VGG16 backbone). Left: balanced accuracy for training and validation. Right: balanced loss for training and validation.

5.2 Robustness

This section reports robustness results. For clarity and space efficiency, we focus on CIFAR-10 as the main benchmark, since it balances complexity and tractability better than MNIST (too simple) and ImageNet-100 (too large-scale for detailed plots). Comprehensive results on MNIST and ImageNet-100 are provided in the Appendix C.2.

5.2.1 Adversarial Attack Results

Figure 5.4 shows adversarial accuracy on CIFAR-10 under FGSM, PGD, and SPGD for $\epsilon \leq 0.3$.

Spiking variants are more robust than ANN across attacks. Adding Flex improves both families; the gains are larger under iterative attacks (PGD, SPGD), where Flex models sustain higher accuracy over a wider ϵ range.

These results suggest complementary effects: temporal sparsity from spiking and adaptive routing from Flex. SNN-BP-Flex is the most robust, indicating that surrogate-gradient training combined with flexible routing better resists gradient-based perturbations.

For reporting clarity, we additionally compute the AUC over $\epsilon \in [0, 0.3]$ (shown below each plot), which provides a compact robustness measure consistent with the curve trends.

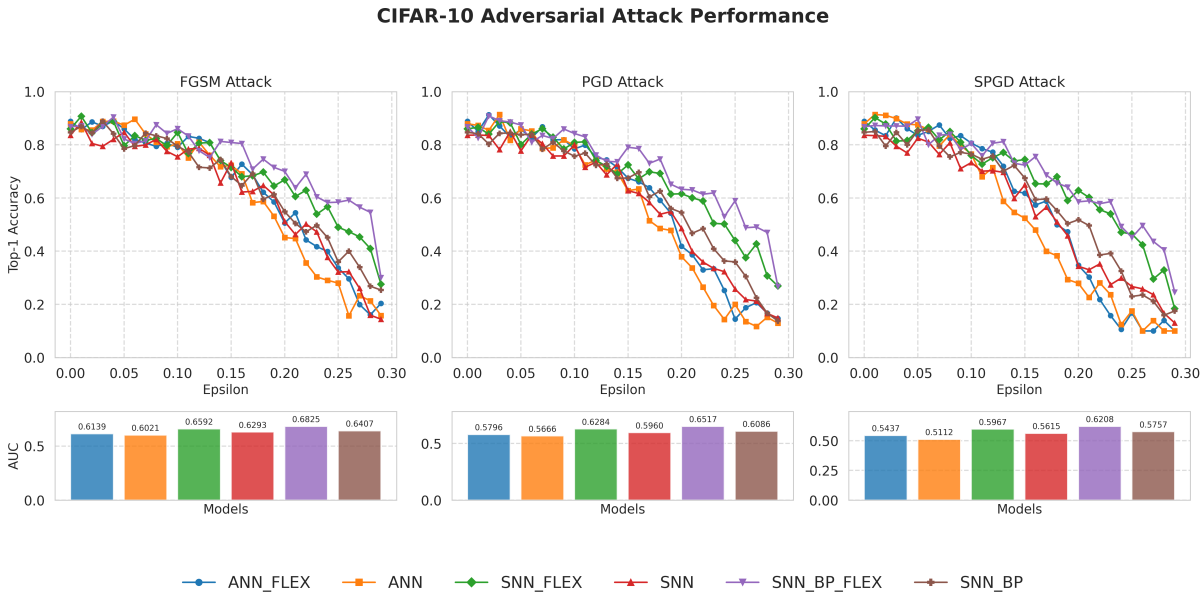


Figure 5.4 CIFAR-10 adversarial robustness. Top: accuracy under FGSM, PGD, and SPGD attacks with increasing perturbation ϵ. Bottom: AUC scores summarising robustness across ϵ. Flex-enhanced SNNs consistently achieve the highest robustness.

5.2.2 Loss Landscape Analysis

To examine stability and generalization, we visualized the loss surfaces of different variants along two random parameter directions (Figure X).

The ANN baseline (Figure 5.5a) shows a broad basin but also sharp cliffs, revealing sensitivity in certain directions. Introducing Flex (Figure 5.5b) yields a smoother surface with gentler slopes, suggesting convergence to flatter minima.

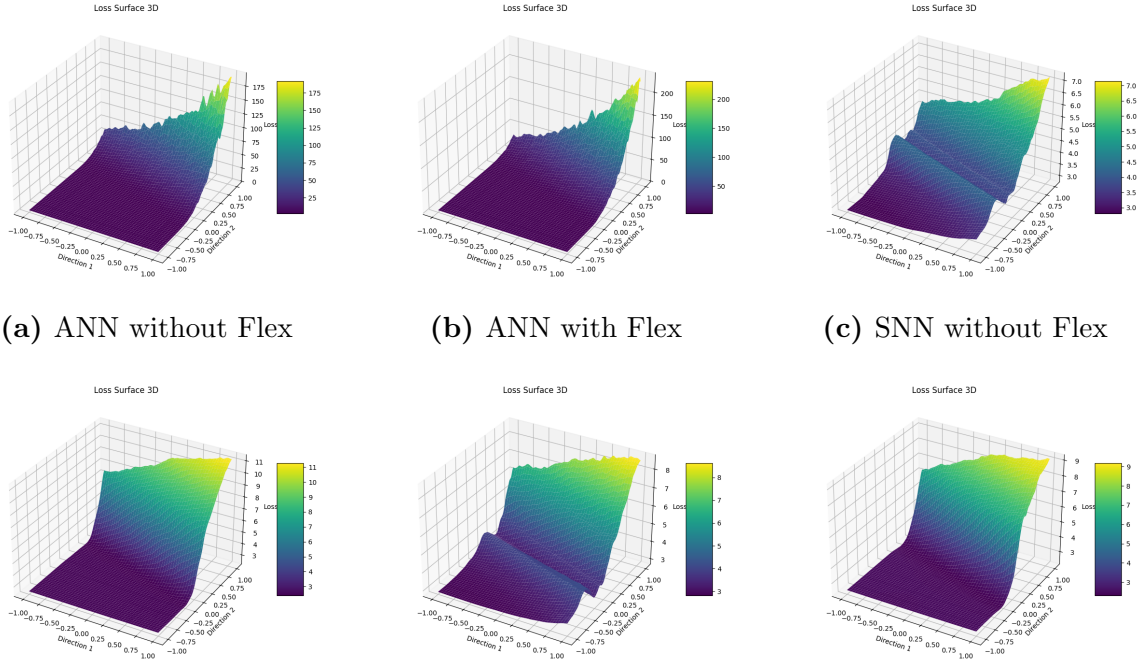
For SNNs, the basic model (Figure 5.5c) exhibits ridge-like structures, indicating uneven sensitivity. Flex reduces these ridges (Figure 5.5d), while backpropagation-through-time (SNN+BP, Figure 5.5e) further regularizes the surface. The combination of BP and Flex (Figure 5.5f) produces the flattest and most stable basin among spiking models.

In general, Flex consistently promotes flatter landscapes in both ANN and SNN settings, and BP training further strengthens this effect in spiking networks.

5.2.3 Hessian Spectrum Analysis

We analysed the Hessian eigenvalue distributions of trained models to characterise local curvature (Figure 5.6).

For ANN baselines (Figure 5.6a), the spectrum spans a wide range with long negative tails, reflecting sharp directions. Introducing Flex (Figure 5.6b) contracts the distribution



(a) ANN without Flex **(b)** ANN with Flex **(c)** SNN without Flex

(d) SNN with Flex **(e)** SNN+BP without Flex **(f)** SNN+BP with Flex

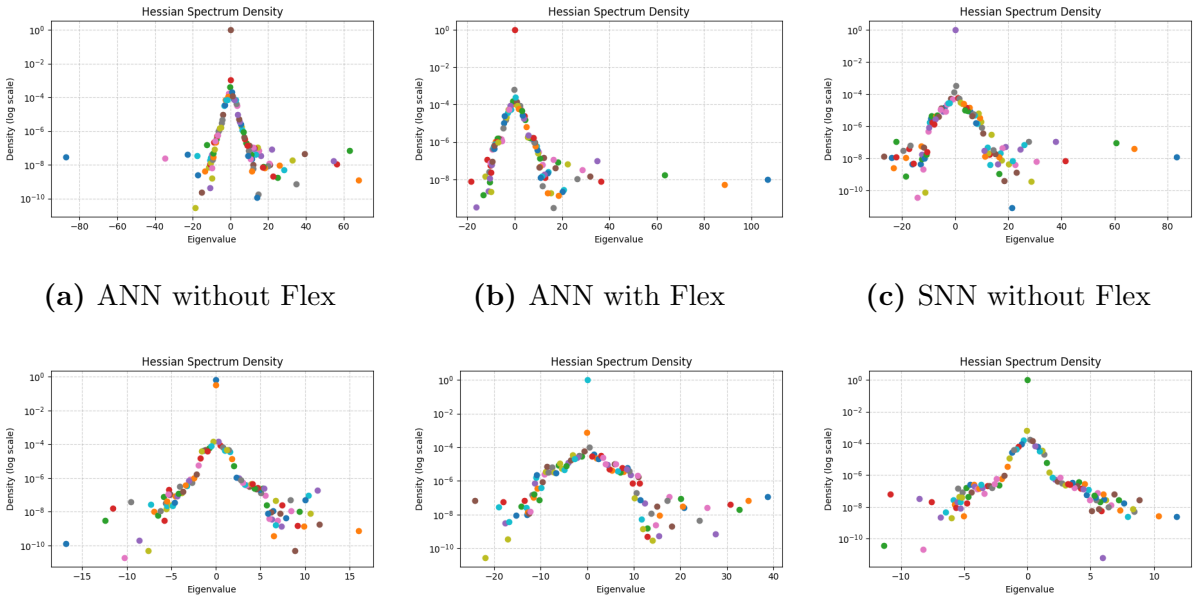
Figure 5.5 Comparison of loss landscapes across six model variants on CIFAR-10 dataset.

toward zero, alleviating part of this instability.

In spiking networks, the plain SNN (Figure 5.6c) exhibits extended positive and negative tails, consistent with rugged curvature. Flex reduces this spread (Figure 5.6d), shifting most eigenvalues closer to zero. Backpropagation training (Figure 5.6e) further suppresses the negative tail, and its combination with Flex (Figure 5.6f) yields the most compact spectrum, with eigenvalues tightly clustered around zero. Flex consistently dampens extreme eigenvalues across both ANN and SNN families, while backpropagation provides an additional stabilizing effect in spiking models.

5.3 Interpretability of Flex Routing

This section reports interpretability analyses of Flex routing. Due to space constraints, we present results on CIFAR-10; full results on MNIST and ImageNet-100 are provided in the appendix C.4.



(a) ANN without Flex **(b)** ANN with Flex **(c)** SNN without Flex

(d) SNN with Flex **(e)** SNN+BP without Flex **(f)** SNN+BP with Flex

Figure 5.6 Hessian spectrum density of six model variants on CIFAR-10 dataset.

5.3.1 Binariness

Figure 5.7 compares the temporal evolution of activation binariness across ANN and SNN variants. In ANN_FLEX, binariness stabilizes around 0.22 with limited variation across layers, indicating that Flex layers in ANN mainly serve structural adjustment without amplifying sparsity effects. In contrast, both SNN_FLEX and SNN_BP_FLEX exhibit higher binariness ($\sim$0.25–0.30), consistent with the spike-driven nature of SNNs. Notably, SNN_FLEX shows stronger divergence between layers, with several deeper layers maintaining binariness above 0.28, whereas SNN_BP_FLEX presents a more balanced distribution, suggesting that backpropagation mitigates extreme sparsification. Across all models, a hierarchical trend emerges: early layers retain lower binariness, while later layers gradually increase, in agreement with the biological selectivity–invariance principle, where shallow stages encode detailed selectivity and deeper stages abstract sparse, invariant representations.

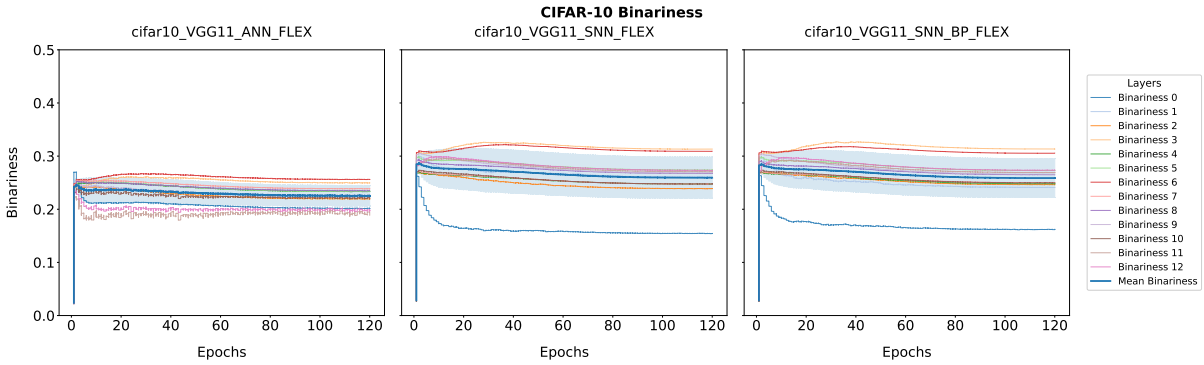


Figure 5.7 Binariness evolution on CIFAR-10 across different architectures.

5.3.2 Conv Ratio–Binariness Evolution and Correlation

Figure 5.8, Figure 5.9, Figure 5.10 compare the evolution of convolutional ratio (Conv Ratio) and activation binariness across ANN and SNN models. Several consistent patterns emerge. First, layer-wise trajectories reveal that early layers converge to higher Conv Ratios, while deeper layers stabilize at lower values. This hierarchical separation is in line with the biological principle of selectivity–invariance, where early stages prioritize feature selectivity and later stages promote invariant abstraction. Second, in the global trend, ANN_FLEX maintains an overall Conv Ratio around 0.4 with stable binariness near 0.23, and no clear coupling is observed (Pearson $r \approx 0.015$). By contrast, SNN models exhibit higher Conv Ratios ($\sim$ 0.5–0.6) together with increased binariness ($\sim$ 0.26–0.30). Notably, both SNN_BP_FLEX and SNN_FLEX show negative correlations between Conv Ratio and binariness (Pearson $r = -0.372$ and $r = -0.200$, respectively), suggesting that higher convolutional dominance suppresses binary activation patterns. Taken together, these results indicate that while Flex layers in ANN primarily act as structural regulators, in SNNs they additionally mediate the balance between convolutional processing and spike-based sparsity, with BP training accentuating this coupling effect.

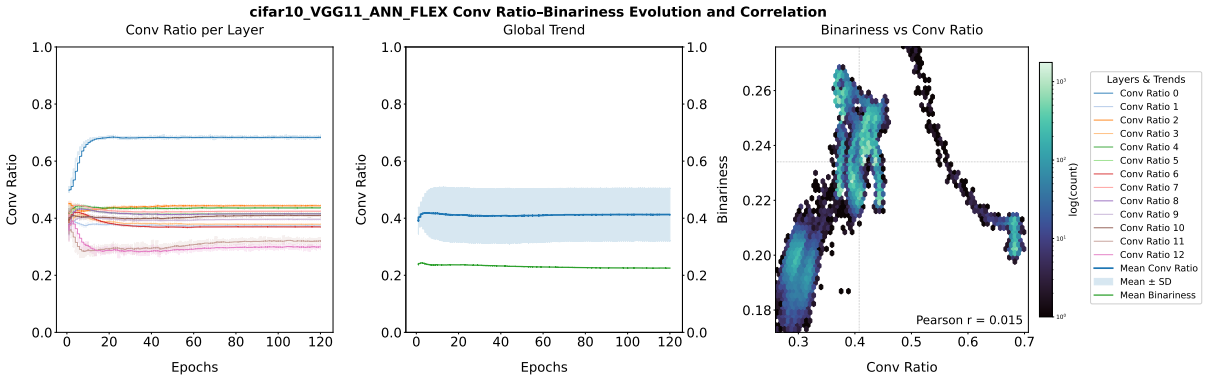


Figure 5.8 CIFAR10-VGG11-ANN-FLEX Conv Ratio–Binariness Dashboard. The left panel shows the evolution of convolution ratios across layers, the middle panel illustrates the global trend with mean ± SD and mean binariness, and the right panel depicts the correlation between binariness and convolution ratio.

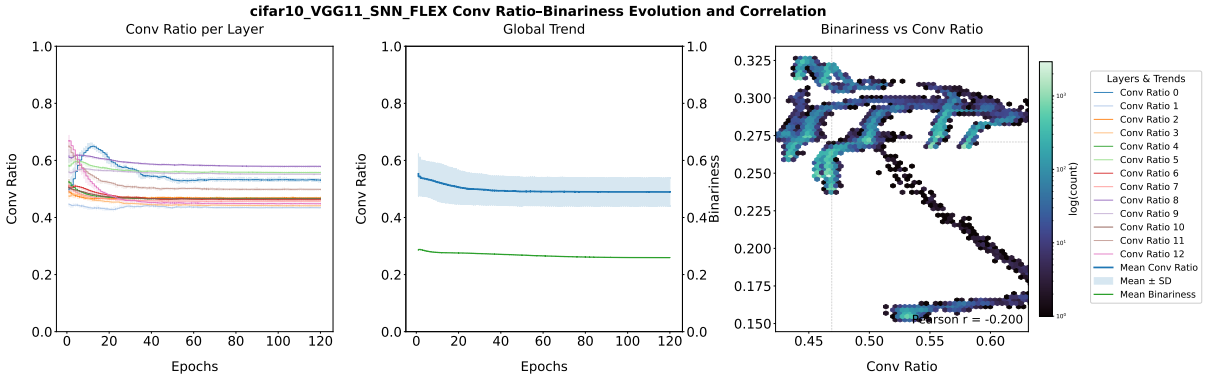


Figure 5.9 CIFAR10-VGG11-SNN-FLEX Conv Ratio–Binariness Dashboard.

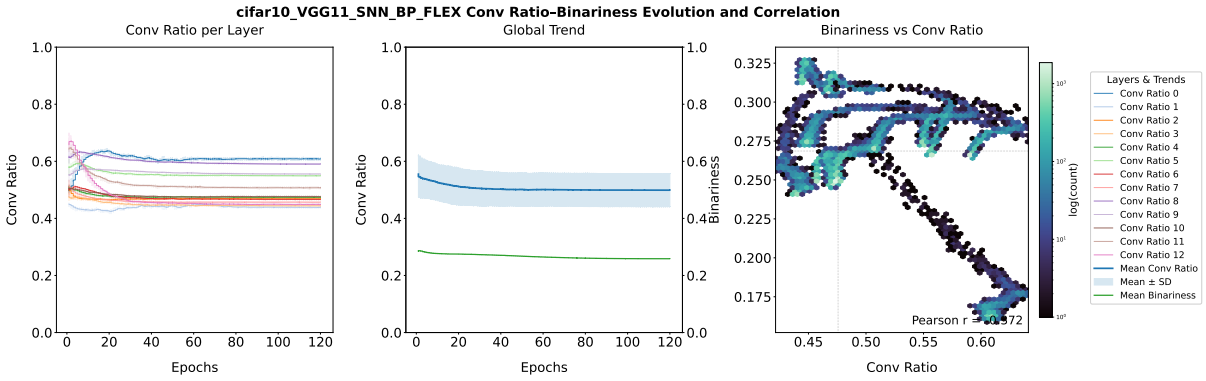
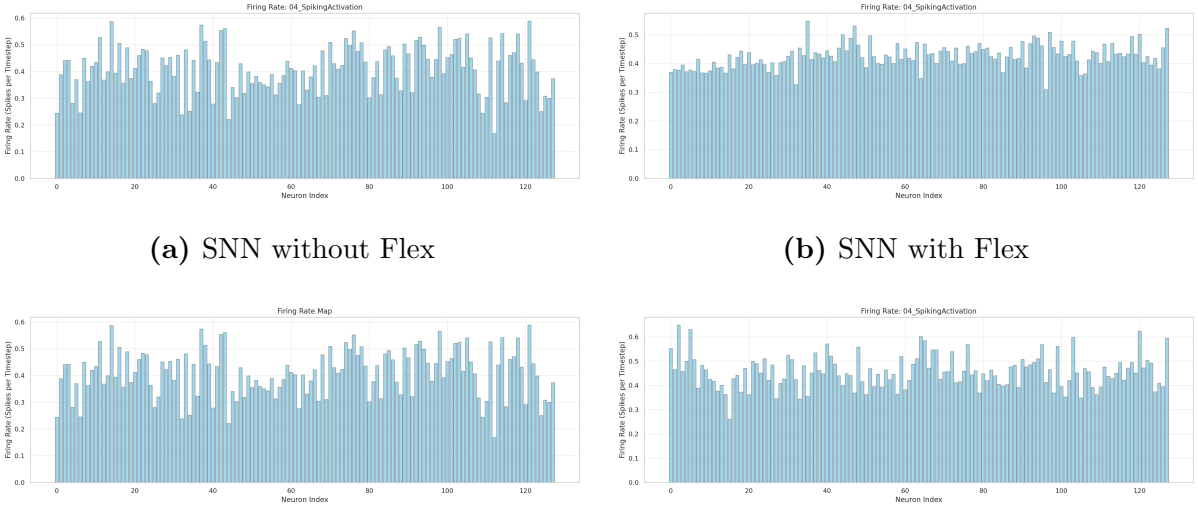


Figure 5.10 CIFAR10-SNN-BP-FLEX Conv Ratio–Binariness Dashboard.

5.4 Spiking Dynamics and Sparsity

5.4.1 Firing Rate Analysis

Figure 5.11 shows firing rate distributions across spiking variants. The plain SNN (Figure 5.11a) displays broad variability, while adding Flex (Figure 5.11b) narrows the spread. With backpropagation (Figure 5.11c), overall firing shifts upward and the tail extends to higher rates. When Flex is combined with BP (Figure 5.11d), the distribution remains elevated but more regularised, indicating that Flex reduces variability in firing while BP increases average activity, and their combination achieves higher yet more controlled firing.



(a) SNN without Flex

(b) SNN with Flex

(c) SNN+BP without Flex

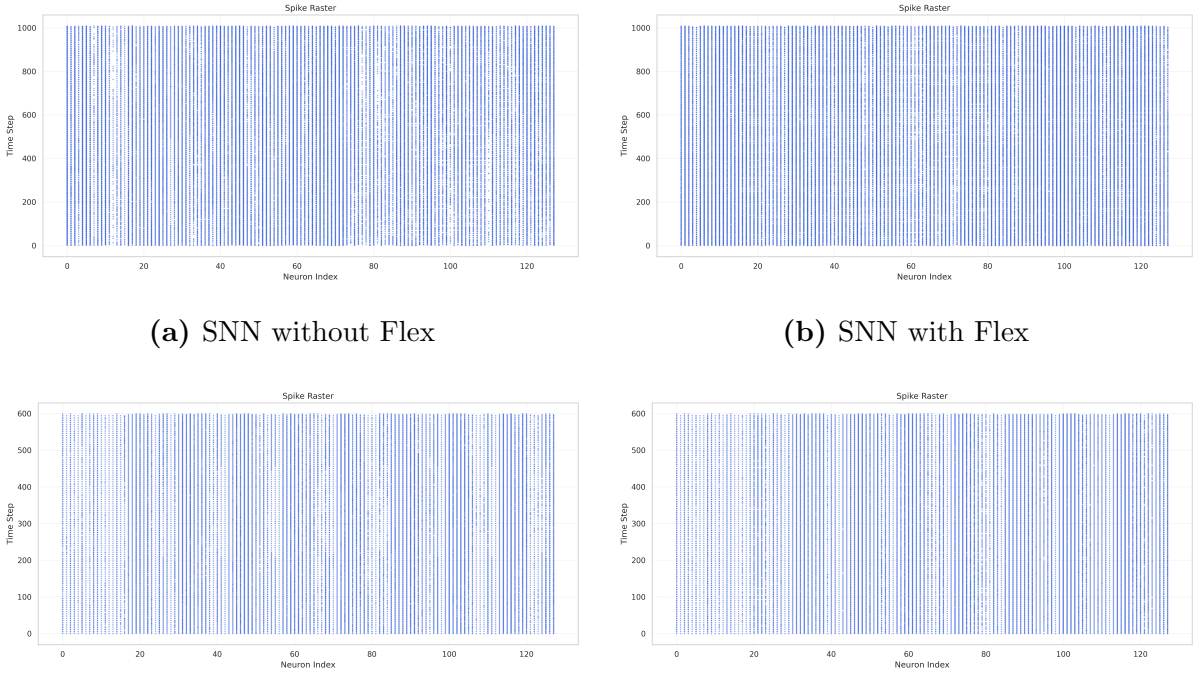
(d) SNN+BP with Flex

Figure 5.11 Firing rate distributions of four spiking model variants.

5.4.2 Raster Plot Analysis

We examined the spiking behavior of different network variants through raster plots. Figure 5.12 illustrates the spike activity across layers for four representative models. The baseline SNN without backpropagation (Figure 5.12a) exhibits dense firing with limited sparsity, while adding the Flex mechanism (Figure 5.12b) produces smoother and sparser activity across layers. When backpropagation with surrogate gradients is applied (Figure 5.12c), firing activity is heavily suppressed, with deeper layers showing almost no spikes. Introducing Flex in this setting (Figure 5.12d) restores a more balanced level of spiking, resulting in a structured and stable distribution across layers. These

observations suggest that Flex reduces redundant spikes and stabilises sparsity, whereas backpropagation alone tends to over-suppress activity.



(a) SNN without Flex

(b) SNN with Flex

(c) SNN+BP without Flex

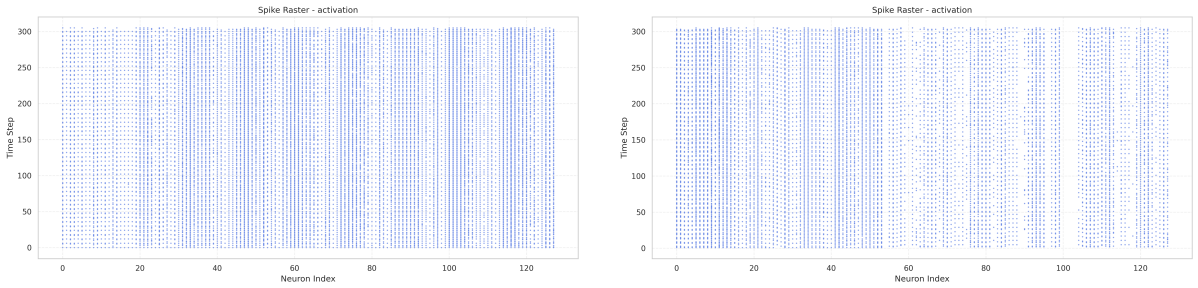
(d) SNN+BP with Flex

Figure 5.12 Raster plots of spike activity across layers. Flex reduces redundant firing and promotes structured sparsity, while backpropagation suppresses spikes more aggressively; combining both achieves a more balanced activity pattern.

To further examine layer-specific dynamics in more detail, we focused on the first convolutional layer of two models: SNN+BP without Flex and SNN+BP with Flex (Figure 5.13). This layer was chosen because it directly encodes input features and largely shapes the network's spiking behaviour. In the SNN+BP model without Flex, the raster plot shows relatively uniform firing across neurons, suggesting limited differentiation. By contrast, the SNN+BP with Flex variant exhibits a distinct "patchy" pattern, where certain neurons are highly active while others remain silent. This selective activation indicates that Flex not only regulates sparsity but also enhances functional differentiation within layers, potentially improving representational efficiency.

5.4.3 Temporal Sparsity Dynamics

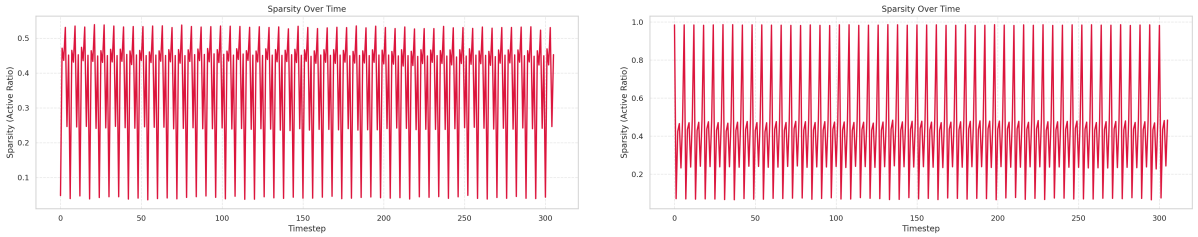
To analyse how Flex affects temporal behaviour under surrogate backpropagation, we compared the evolution of activity sparsity between SNN+BP and SNN+BP with Flex (Figure 5.14). The baseline SNN+BP maintains a relatively stable active ratio around



(a) SNN+BP without Flex **(b)** SNN+BP with Flex

Figure 5.13 Raster plots of the first convolutional layer. Without Flex, neurons fire uniformly with little differentiation, whereas Flex induces a patchy pattern with selective activation, highlighting its role in promoting functional diversity.

0.4–0.5 across timesteps, indicating a strong global suppression that keeps firing sparse yet uniform. By contrast, the Flex-augmented model displays pronounced oscillations between near-silent and highly active states, suggesting that mask routing introduces temporal modulation of activity. This shows that Flex does not merely reduce spiking overall, but redistributes events selectively, creating a richer temporal structure in the activations.



(a) SNN+BP without Flex **(b)** SNN+BP with Flex

Figure 5.14 Temporal sparsity dynamics under surrogate backpropagation. Without Flex, firing remains uniformly sparse across time, while Flex introduces oscillatory modulation, alternating between suppressed and highly active states.

Chapter 6

Discussion

6.1 Analysis of Results

Across datasets and evaluation dimensions, the results consistently show that the proposed Flex mechanism enhances both stability and robustness without sacrificing accuracy. From a biological perspective, this effect can be linked to the principle of selectivity–invariance: early layers maintain higher convolutional ratios, preserving detailed selectivity of local features, while deeper layers favour pooling, promoting invariant abstraction. This hierarchical shift mirrors cortical processing, where early sensory areas encode fine-grained information and higher areas compress it into sparse, invariant codes.

The binariness and firing analyses further suggest that Flex acts as a regulator of sparsity. In artificial neural networks, Flex maintains moderate binariness, mainly shaping structural balance between convolution and pooling. In spiking networks, however, Flex interacts with spike-driven dynamics to modulate both spatial and temporal sparsity, reducing redundant firing while preserving discriminative signals. This behaviour reflects a form of efficient coding: by redistributing activity across neurons and timesteps, the network approximates information-theoretic principles such as entropy maximisation and redundancy reduction.

Loss landscape and Hessian analyses provide a complementary view. Models equipped with Flex consistently converge to flatter minima with spectra concentrated near zero, indicating reduced sensitivity to perturbations. In information-theoretic terms, such flatter solutions correspond to models that generalise by avoiding sharp decision boundaries, effectively trading representational capacity for robustness against input and parameter noise. The surrogate-gradient trained SNNs with Flex show the most compact spectra, suggesting that combining temporal sparsity with adaptive routing yields particularly stable solutions.

Taken together, these findings highlight that the Flex mechanism not only improves classification and robustness in engineering terms, but also enforces constraints that resemble biological efficiency principles: selective routing of information, sparse and temporally modulated activity, and convergence to flatter, more stable representations. This

alignment between computational performance and biological plausibility suggests that Flex-style adaptive modules may provide a principled way to bridge artificial and spiking neural systems under a unified framework.

6.2 Neuroscience Connection

The Flex routing mechanism parallels the role of cortical microcircuits that dynamically gate inputs through recurrent and feedforward pathways. Mask binariness reflects a process similar to synaptic switching, where local circuits stabilise into discrete routing states under repeated stimulation. The observed trade-off between convolutional ratio and binariness aligns with the balance of excitation and inhibition in cortical layers, which regulates both selectivity and sparsity. Temporal modulation of spiking activity resembles oscillatory gating in thalamocortical loops, where bursts and silences structure information flow across time. The flatter loss surfaces found in Flex models echo the stability of attractor states in recurrent cortical networks, supporting reliable computation under noisy conditions. Finally, the efficiency gains from reduced firing variability reflect an information-theoretic principle also seen in sensory coding, where redundancy is minimised while maintaining discriminability.

Chapter 7

Conclusions and Future Work

7.1 Summary and Conclusions

This thesis introduced *PlasticFlex*, a unified framework that extends conventional convolutional networks to support both ANN and SNN modes, while enabling adaptive routing through the Flex2D module. By replacing standard convolution layers with a dual-branch mechanism—pooling and stride-2 convolution combined via a learnable mask—the network was able to regulate the balance between information preservation and abstraction in a data-driven manner.

Experiments across MNIST, CIFAR-10, and ImageNet-100 confirmed that introducing Flex did not compromise classification accuracy. In fact, modest but consistent improvements were observed in both ANN and SNN settings, indicating that adaptive routing enhanced discriminative capacity without adding significant architectural complexity. More importantly, spiking models incorporating Flex achieved the highest robustness under gradient-based adversarial attacks, suggesting that temporal sparsity and spatially adaptive routing provide complementary defenses against perturbations.

Analysis of loss landscapes and Hessian spectra showed that Flex systematically smooths optimization surfaces, producing flatter minima and reducing the dominance of extreme eigenvalues. These effects were observed in both ANN and SNN families, but were most pronounced when Flex was combined with surrogate-gradient backpropagation in spiking models. Such stabilization was further reflected in firing rate distributions and raster plots: Flex regularised neuronal activity by reducing variability and encouraging structured sparsity, while backpropagation alone tended to over-suppress spiking.

Interpretability analyses revealed consistent layer-wise patterns: early layers converged to higher convolution ratios and lower binariness, while later layers became increasingly sparse, aligning with the biological selectivity–invariance principle. The negative correlation between convolution dominance and binariness in SNNs further suggested that Flex mediates a trade-off between precise feature extraction and efficient spike-based abstraction.

Taken together, these findings establish PlasticFlex as a framework that integrates

adaptive routing, spiking dynamics, and surrogate learning into a coherent design. Beyond achieving competitive accuracy, the framework provides robustness, interpretability, and biologically grounded sparsity, pointing towards a principled pathway for developing neural architectures that balance efficiency, stability, and expressive power.

7.2 Limitations and Future Works

While the proposed PlasticFlex framework demonstrates consistent improvements in robustness, interpretability, and biological plausibility, several limitations remain. First, all evaluations were performed on image classification benchmarks. The extent to which adaptive routing generalises to sequential tasks such as speech decoding, reinforcement learning, or neuromorphic sensing remains unexplored. Second, the Flex2D module was applied uniformly across all convolutional layers; more selective placement or hierarchical parameterisation might improve efficiency without increasing model size. Third, although surrogate-gradient training enabled effective optimisation in spiking models, its biological plausibility and scalability to very deep SNNs remain open questions. Fourth, the current evaluation of sparsity and efficiency was limited to firing rate statistics and mask binarisation; a full account of energy efficiency would require deployment on neuromorphic hardware or detailed computational cost modelling. Finally, while Flex offered interpretable statistics such as convolution ratio and binariness, linking these measures directly to cognitive-level phenomena (e.g., invariance formation in perception) will require joint analysis with neural recordings.

Future work may therefore extend PlasticFlex in three complementary directions: task generalisation beyond static vision, architectural refinement through adaptive placement of routing modules, and neuromorphic realisation for energy-constrained hardware. In addition, deeper integration with neuroscience could test whether routing dynamics observed in Flex correspond to strategies employed by biological circuits, bridging algorithmic mechanisms with neural computation in a more principled manner.

Appendix A

Theorical Analysis

Goal: To demonstrate that the PlasticFlex Layer maintains stability, interpretability, and certifiability despite its structural plasticity.

A.1 Certifiability

Step 1: Layer Definition The PlasticFlex layer computes the output at each location $i \in \{1, \ldots, N\}$ as:

$$y_i = m_i \cdot C_i + (1 - m_i) \cdot P_i \tag{A.1}$$

where C_i and P_i denote the outputs of the convolutional and pooling paths, respectively, and $m_i \in [0, 1]$ is a routing mask.

We assume that each component lies within known bounds:

$$C_i \in [l_c,\ u_c] \tag{A.2}$$

$$P_i \in [l_p,\ u_p] \tag{A.3}$$

$$m_i \in [l_m,\ u_m] \subseteq [0, 1] \tag{A.4}$$

Step 2: Affine Expansion We rewrite Equation (A.1) to expose the affine structure:

$$y_i = m_i(C_i - P_i) + P_i \tag{A.5}$$

Let us define a function:

$$f(m, C, P) := m(C - P) + P$$

Our goal is to compute:

$$\min_{(m,C,P)\in[l_m,u_m]\times[l_c,u_c]\times[l_p,u_p]} f(m, C, P) \quad \text{and} \quad \max f(m, C, P)$$

Step 3: Domain Geometry The domain is a closed box (parallelepiped) in $\mathbb{R}^3$, with 8 vertices given by:

$$(m, C, P) \in \{l_m, u_m\} \times \{l_c, u_c\} \times \{l_p, u_p\}$$

The function $f(m, C, P)$ is continuous and piecewise linear. Therefore, its global minimum and maximum over a box domain are attained at one of the 8 vertices.

Step 4: Enumerate All 8 Vertices Let us write out all 8 corner values of f:

$$\begin{aligned}
v_1 &= f(l_m, l_c, l_p) = l_m(l_c - l_p) + l_p \\
v_2 &= f(u_m, l_c, l_p) = u_m(l_c - l_p) + l_p \\
v_3 &= f(l_m, u_c, l_p) = l_m(u_c - l_p) + l_p \\
v_4 &= f(u_m, u_c, l_p) = u_m(u_c - l_p) + l_p \\
v_5 &= f(l_m, l_c, u_p) = l_m(l_c - u_p) + u_p \\
v_6 &= f(u_m, l_c, u_p) = u_m(l_c - u_p) + u_p \\
v_7 &= f(l_m, u_c, u_p) = l_m(u_c - u_p) + u_p \\
v_8 &= f(u_m, u_c, u_p) = u_m(u_c - u_p) + u_p
\end{aligned}$$

Step 5: Collect Bounds from Evaluations The output is bounded as:

$$y_i \in \left[\min_{1 \le j \le 8} v_j, \quad \max_{1 \le j \le 8} v_j \right]$$

While this bound is exact, evaluating 8 combinations per unit is costly. Fortunately, we can reduce it to 4 cases using structural symmetry.

Step 6: Reduce to 4 Evaluations Observe that f is affine in all variables. Its extrema must lie at one of the 8 vertices, but for convex interpolation (Eq. (A.1)), only the combinations where C_i and P_i are both at either their lower or upper bounds affect the extremes significantly.

Thus, we define:

$$\mathcal{E}_1 = l_m \cdot l_c + (1 - l_m) \cdot l_p \tag{A.6}$$

$$\mathcal{E}_2 = u_m \cdot l_c + (1 - u_m) \cdot l_p \tag{A.7}$$

$$\mathcal{E}_3 = l_m \cdot u_c + (1 - l_m) \cdot u_p \tag{A.8}$$

$$\mathcal{E}_4 = u_m \cdot u_c + (1 - u_m) \cdot u_p \tag{A.9}$$

Step 7: Final Tight Bound Expression

$$y_i \in [\min(\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4),\ \max(\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4)] \tag{A.10}$$

This expression tightly encloses all 8 vertices while requiring only 4 evaluations.

Step 8: Vectorization For a tensor-valued input, this bound generalizes element-wise as:

$$\boxed{\mathbf{y} \in [\min(\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4),\ \max(\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4)]} \tag{A.11}$$

All operations are applied element-wise across dimensions (C, H, W).

Conclusion. The PlasticFlex operator admits tight, closed-form certifiable output bounds under independent interval assumptions on its inputs. This confirms its compatibility with interval-based verification frameworks and its safety under bounded input perturbations.

A.2 Interval Bound Propagation

Motivation. To establish the certifiability of the PlasticFlex layer (Appendix A.1), we require a tight characterization of how bounded input perturbations propagate through the layer. In this section, we derive explicit upper and lower bounds for the PlasticFlex output, assuming that each of its inputs lies within a known interval.

Step 1: Layer Output Definition. The PlasticFlex output at spatial location $i \in \{1, \ldots, N\}$ is:

$$y_i = m_i \cdot C_i + (1 - m_i) \cdot P_i \tag{A.12}$$

where:

- m_i is the mask value at location i,
- C_i and P_i are the outputs from the convolution and pooling branches, respectively.

Step 2: Interval Assumptions. We assume the following intervals are given:

$$C_i \in [l_c,\ u_c] \tag{A.13}$$

$$P_i \in [l_p,\ u_p] \tag{A.14}$$

$$m_i \in [l_m,\ u_m] \subseteq [0, 1] \tag{A.15}$$

Our goal is to compute a tight enclosing interval $[l_y,\ u_y]$ such that:

$$y_i \in [l_y,\ u_y]$$

Step 3: Reformulation as a Function. Let us define the scalar function:

$$f(m, C, P) = m(C - P) + P$$

Then,

$$y_i = f(m_i, C_i, P_i)$$

We wish to compute:

$$\min_{m\in[l_m,u_m],\ C\in[l_c,u_c],\ P\in[l_p,u_p]} f(m, C, P) \quad \text{and} \quad \max f(m, C, P)$$

Step 4: Structure of $f(m, C, P)$. We analyze the function:

$$f(m, C, P) = m(C - P) + P$$

Observe: - f is affine in m, C, and P. - f is continuous and differentiable over the box domain $[l_m, u_m] \times [l_c, u_c] \times [l_p, u_p]$. - Therefore, extrema are attained at some vertex of the box.

Step 5: Enumerate All 8 Vertices. The domain has $2^3 = 8$ vertices:

$$(m, C, P) \in \{l_m, u_m\} \times \{l_c, u_c\} \times \{l_p, u_p\}$$

We define:

$$\begin{aligned}
v_1 &= f(l_m, l_c, l_p) = l_m(l_c - l_p) + l_p \\
v_2 &= f(u_m, l_c, l_p) = u_m(l_c - l_p) + l_p \\
v_3 &= f(l_m, u_c, l_p) = l_m(u_c - l_p) + l_p \\
v_4 &= f(u_m, u_c, l_p) = u_m(u_c - l_p) + l_p \\
v_5 &= f(l_m, l_c, u_p) = l_m(l_c - u_p) + u_p \\
v_6 &= f(u_m, l_c, u_p) = u_m(l_c - u_p) + u_p \\
v_7 &= f(l_m, u_c, u_p) = l_m(u_c - u_p) + u_p \\
v_8 &= f(u_m, u_c, u_p) = u_m(u_c - u_p) + u_p
\end{aligned}$$

Step 6: Define Bound as Min/Max over Vertices. The tight interval is:

$$y_i \in \left[\min_{1\le j\le 8} v_j, \quad \max_{1\le j\le 8} v_j\right]$$

Step 7: Simplify to 4 Critical Expressions. We now simplify by noting the output expression in Equation (A.12) is convex in m (as C, P are fixed), and piecewise linear over the domain.

In practice, the extremal values are determined by 4 configurations:

$$\mathcal{E}_1 = l_m \cdot l_c + (1 - l_m) \cdot l_p \tag{A.16}$$

$$\mathcal{E}_2 = u_m \cdot l_c + (1 - u_m) \cdot l_p \tag{A.17}$$

$$\mathcal{E}_3 = l_m \cdot u_c + (1 - l_m) \cdot u_p \tag{A.18}$$

$$\mathcal{E}_4 = u_m \cdot u_c + (1 - u_m) \cdot u_p \tag{A.19}$$

Step 8: Final Interval Bound Expression. The output is tightly enclosed by:

$$y_i \in [\min(\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4),\ \max(\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4)] \tag{A.20}$$

Step 9: Vectorized Form. The above derivation generalizes element-wise for the full tensor $\mathbf{y} \in \mathbb{R}^{C\times H\times W}$:

$$\boxed{\mathbf{y} \in [\min(\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4),\ \max(\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4)]} \tag{A.21}$$

where all operations are element-wise.

Conclusion. We have derived the complete output interval for the PlasticFlex layer under independently bounded inputs. This result enables the safe integration of PlasticFlex into interval-based certified robustness pipelines, and provides the foundation for the certifiability analysis in Certifiability part.

A.3 Mask Sensitivity

Motivation. While Appendix A.1 and A.2 establish that the PlasticFlex layer is certifiably bounded under input perturbations, it remains important to understand the functional role of its learned structural mask m_i. In this section, we quantify how sensitive the layer output is to changes in the mask, and how this sensitivity relates to both interpretability and mask trainability.

Step 1: PlasticFlex Output Definition As before, the PlasticFlex output at each spatial index i is:

$$y_i = m_i \cdot C_i + (1 - m_i) \cdot P_i \tag{A.22}$$

where C_i, P_i are the convolutional and pooling branch outputs, and $m_i \in [0, 1]$ is the routing mask.

Step 2: Sensitivity to Mask – Local Derivative We first compute the local sensitivity of the output to the mask by taking a partial derivative:

$$\begin{aligned} \frac{\partial y_i}{\partial m_i} &= \frac{\partial}{\partial m_i} \left(m_i \cdot C_i + (1 - m_i) \cdot P_i \right) \\ &= C_i - P_i \end{aligned} \tag{A.23}$$

This expression measures how much the output changes per unit change in the routing decision m_i.

Step 3: Define Sensitivity Score We define a local scalar mask sensitivity score as:

$$S_i := \left| \frac{\partial y_i}{\partial m_i} \right| = |C_i - P_i| \tag{A.24}$$

This term quantifies the structural contrast at location i: the greater the difference between the two branches, the more meaningful the routing decision becomes.

Step 4: Global Average Sensitivity To summarize sensitivity across the entire feature map:

$$\bar{S} := \frac{1}{N} \sum_{i=1}^{N} S_i = \frac{1}{N} \sum_{i=1}^{N} |C_i - P_i| \tag{A.25}$$

This global metric reflects the network's overall reliance on flexible structure. Higher $\bar{S}$ indicates greater use of mask-based routing.

Step 5: Backpropagation Gradient into Mask The gradient of the loss $\mathcal{L}$ with respect to m_i is:

$$\frac{\partial \mathcal{L}}{\partial m_i} = \frac{\partial \mathcal{L}}{\partial y_i} \cdot \frac{\partial y_i}{\partial m_i} = \frac{\partial \mathcal{L}}{\partial y_i} \cdot (C_i - P_i) \tag{A.26}$$

Thus, S_i not only measures interpretability, but also affects the trainability of the mask: low contrast leads to vanishing gradients.

Step 6: Perturbation Bound on Output Suppose the mask is perturbed by a small amount: $m_i \to m_i + \delta_i$.

Then:

$$\begin{aligned} y_i' &= (m_i + \delta_i)C_i + (1 - m_i - \delta_i)P_i \\ &= y_i + \delta_i(C_i - P_i) \qquad \text{(A.27)} \\ \Rightarrow \ |\Delta y_i| &= |\delta_i| \cdot |C_i - P_i| = |\delta_i| \cdot S_i \qquad \text{(A.28)} \end{aligned}$$

This bound shows that in high-sensitivity regions (large S_i), small changes in mask can lead to significant output shifts—informing both regularization and robustness.

Step 7: Interpretation and Utility

- S_i captures the functional relevance of the routing mask at location i;
- $\bar{S}$ quantifies the global importance of plasticity in the network;
- Equation (A.26) shows that S_i modulates the learning signal for m_i;
- Equation (A.28) links sensitivity to robustness, motivating potential mask regularization in adversarial settings.

Conclusion. The PlasticFlex layer exhibits interpretable and quantifiable structural sensitivity. This derivation complements the interval-bound stability (Appendix A.2) and certifiability results (Appendix A.1), forming a complete analysis of the layer's reliability and functional plasticity.

Appendix B

Model Architectures and Hyperparameters

B.1 VGG Architecture Tables

This appendix lists the backbone architectures used in our experiments.

B.1.1 VGG-6 for MNIST

Table 2.1 Baseline VGG-6 architecture for MNIST.

Block	Layers
Block 1	Conv(3×3, 64) Conv(3×3, 128) MaxPool(2×2)
Block 2	Conv(3×3, 256) Conv(3×3, 256) MaxPool(2×2)
FC	FC1 (1024 units, Dropout) FC2 (512 units, Dropout) Output (10 classes)

B.1.2 VGG-11 for CIFAR-10

B.1.3 VGG-16 for ImageNet-100

B.2 Training Hyperparameters

B.2.1 MNIST

Shared settings. Unless otherwise noted, all MNIST runs use: VGG6 backbone; SGD with momentum 0.9 and weight decay 5×10^{-4}; cosine learning rate schedule with initial

Table 2.2 Flex-enabled VGG-6 for MNIST (all conv and pooling layers replaced by PlasticFlex).

Block	Layers
Block 1	PlasticFlex(64) PlasticFlex(128, stride 2)
Block 2	PlasticFlex(256) PlasticFlex(256, stride 2)
FC	FC1 (1024 units, Dropout) FC2 (512 units, Dropout) Output (10 classes)

Table 2.3 Baseline VGG-11 architecture for CIFAR-10.

Block	Layers
Block 1	Conv(3×3, 64) MaxPool(2×2)
Block 2	Conv(3×3, 128) MaxPool(2×2)
Block 3	Conv(3×3, 256) Conv(3×3, 256) MaxPool(2×2)
Block 4	Conv(3×3, 512) Conv(3×3, 512) MaxPool(2×2)
Block 5	Conv(3×3, 512) Conv(3×3, 512) MaxPool(2×2)
FC	FC1 (2048 units, Dropout) FC2 (1024 units, Dropout) Output (10 classes)

Table 2.4 Flex-enabled VGG-11 for CIFAR-10.

Block	Layers
Block 1	PlasticFlex(64, stride 2)
Block 2	PlasticFlex(128, stride 2)
Block 3	PlasticFlex(256) PlasticFlex(256, stride 2)
Block 4	PlasticFlex(512) PlasticFlex(512, stride 2)
Block 5	PlasticFlex(512) PlasticFlex(512, stride 2)
FC	FC1 (2048 units, Dropout) FC2 (1024 units, Dropout) Output (10 classes)

rate 0.1; batch size 256; dropout $p = 0.3$ after each fully connected layer; mask parameterisation: hard-sigmoid; logits mechanism: threshold; logits BatchNorm: enabled; spiking threshold 1.0 and membrane decay 0.95 (when spiking is enabled).

B.2.2 CIFAR-10

Shared settings. Unless otherwise noted, all CIFAR-10 runs use: VGG11 backbone; SGD with momentum 0.9 and weight decay 5×10^{-4}; cosine learning rate schedule with initial rate 0.1; batch size 64; dropout $p = 0.2$ after each fully connected layer; mask parameterisation: hard-sigmoid; logits mechanism: threshold; logits BatchNorm: enabled; spiking threshold 1.0 and membrane decay 0.95 (when spiking is enabled).

B.2.3 ImageNet-100

Shared settings. Unless otherwise noted, all ImageNet-100 runs use: VGG16 backbone; SGD with momentum 0.9 and weight decay 5×10^{-4}; cosine learning rate schedule with initial rate 0.1; batch size 256; dropout $p = 0.2$ after each fully connected layer; mask parameterisation: hard-sigmoid; logits mechanism: threshold; logits BatchNorm: enabled; spiking threshold 1.0 and membrane decay 0.95 (when spiking is enabled).

Table 2.5 Baseline VGG-16 architecture for ImageNet-100.

Block	Layers
Block 1	Conv(3×3, 64) Conv(3×3, 64) MaxPool(2×2)
Block 2	Conv(3×3, 128) Conv(3×3, 128) MaxPool(2×2)
Block 3	Conv(3×3, 256) Conv(3×3, 256) Conv(3×3, 256) MaxPool(2×2)
Block 4	Conv(3×3, 512) Conv(3×3, 512) Conv(3×3, 512) MaxPool(2×2)
Block 5	Conv(3×3, 512) Conv(3×3, 512) Conv(3×3, 512) MaxPool(2×2)
FC	FC1 (4096 units, Dropout) FC2 (2048 units, Dropout) Output (100 classes)

Table 2.6 Flex-enabled VGG-16 for ImageNet-100.

Block	Layers
Block 1	PlasticFlex(64) PlasticFlex(64, stride 2)
Block 2	PlasticFlex(128) PlasticFlex(128, stride 2)
Block 3	PlasticFlex(256) PlasticFlex(256) PlasticFlex(256, stride 2)
Block 4	PlasticFlex(512) PlasticFlex(512) PlasticFlex(512, stride 2)
Block 5	PlasticFlex(512) PlasticFlex(512) PlasticFlex(512, stride 2)
FC	FC1 (4096 units, Dropout) FC2 (2048 units, Dropout) Output (100 classes)

Table 2.7 MNIST variants and differing hyperparameters. Abbreviations: BP (surrogate backprop).

Variant	Flex	SNN	BP	Epochs	Batch size	Init LR	spiking threshold	membrane decay
ANN (baseline)	–	–	–	120	128	0.10	–	–
ANN+Flex	✓	–	–	120	128	0.10	–	–
SNN	–	✓	–	120	128	0.10	1.0	0.95
SNN+Flex	✓	✓	–	120	128	0.10	1.0	0.95
SNN + BP	–	✓	✓	120	128	0.10	1.0	0.95
SNN + BP+Flex	✓	✓	✓	120	128	0.10	1.0	0.95

Table 2.8 CIFAR-10 variants and differing hyperparameters. Abbreviations: BP (surrogate backprop).

Variant	Flex	SNN	BP	Epochs	Batch size	Init LR	spiking threshold	membrane decay
ANN (baseline)	–	–	–	120	128	0.10	–	–
ANN+Flex	✓	–	–	120	128	0.10	–	–
SNN	–	✓	–	120	128	0.10	1.0	0.95
SNN+Flex	✓	✓	–	120	128	0.10	1.0	0.95
SNN + BP	–	✓	✓	120	128	0.10	1.0	0.95
SNN + BP+Flex	✓	✓	✓	120	128	0.10	1.0	0.95

Table 2.9 ImageNet-100 variants and differing hyperparameters. Abbreviations: BP (surrogate backprop).

Variant	Flex	SNN	BP	Epochs	Batch size	Init LR	spiking threshold	membrane decay
ANN (baseline)	–	–	–	180	256	0.10	–	–
ANN+Flex	✓	–	–	180	256	0.10	–	–
SNN	–	✓	–	180	256	0.10	1.0	0.95
SNN+Flex	✓	✓	–	180	256	0.10	1.0	0.95
SNN + BP	–	✓	✓	180	256	0.10	1.0	0.95
SNN + BP+Flex	✓	✓	✓	180	256	0.10	1.0	0.95

Appendix C

Supplementary Experimental Results

C.1 Top-1 Classification Accuracy Results

The following figures report the Top-1 accuracy results on MNIST, CIFAR-10, and ImageNet-100.

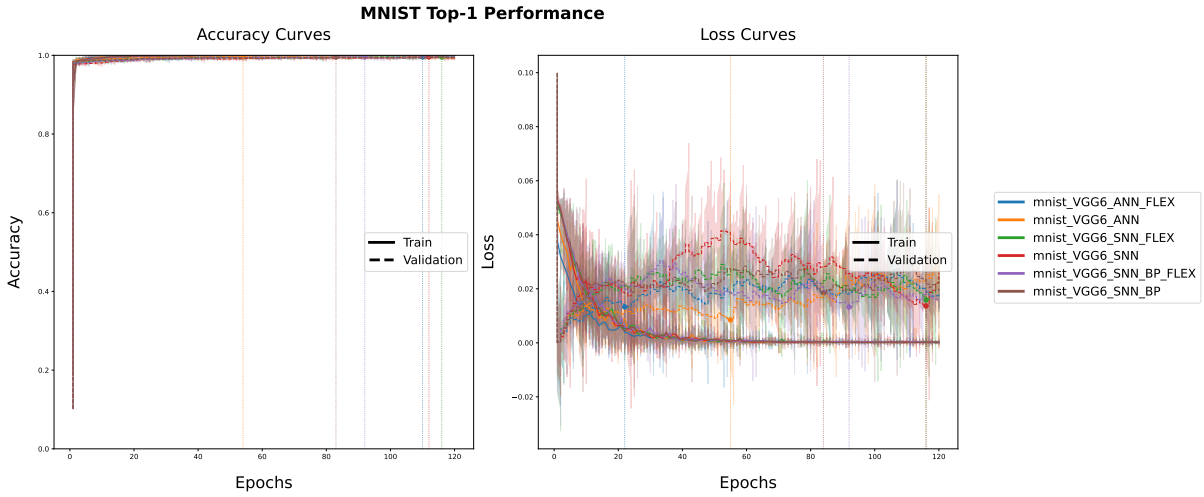


Figure 3.1 MNIST top-1 training curves (VGG16 backbone).

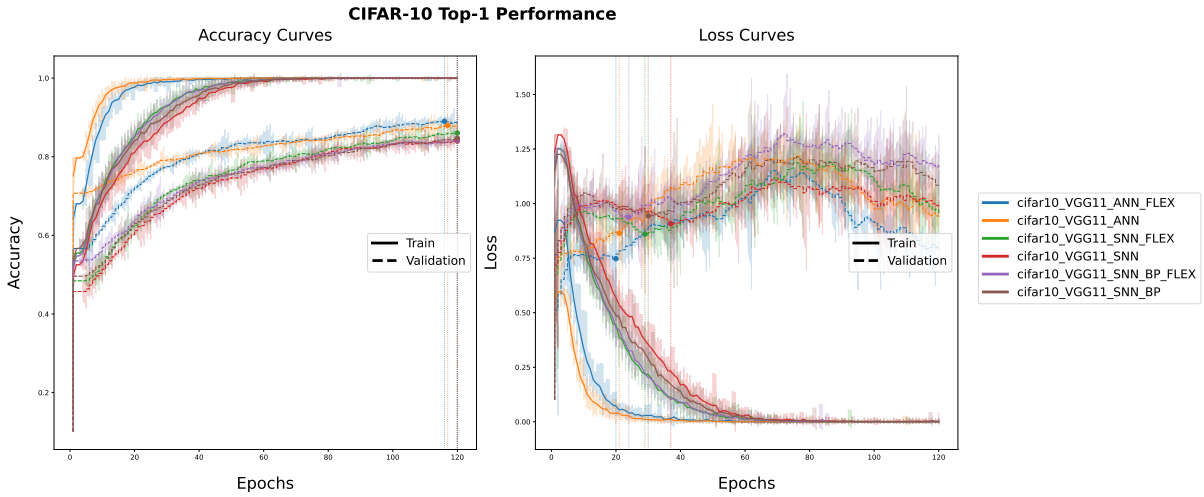


Figure 3.2 CIFAR-10 top-1 training curves (VGG16 backbone).

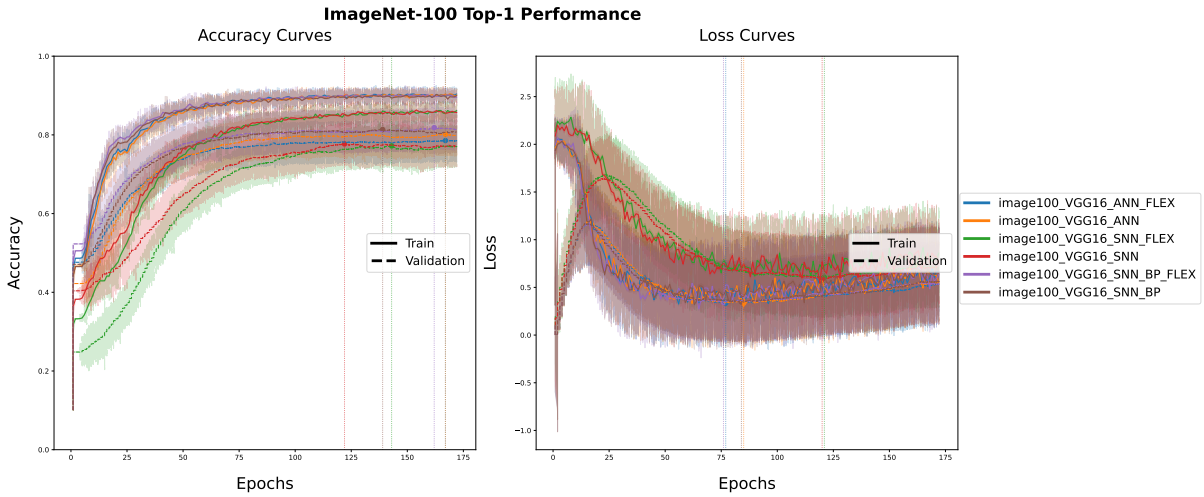


Figure 3.3 ImageNet-100 top-1 training curves (VGG16 backbone).

C.2 Additional Robustness Results

C.2.1 Adversarial Attacks

The following figures report the adverserial attack results on MNIST, CIFAR-10, and ImageNet-100.

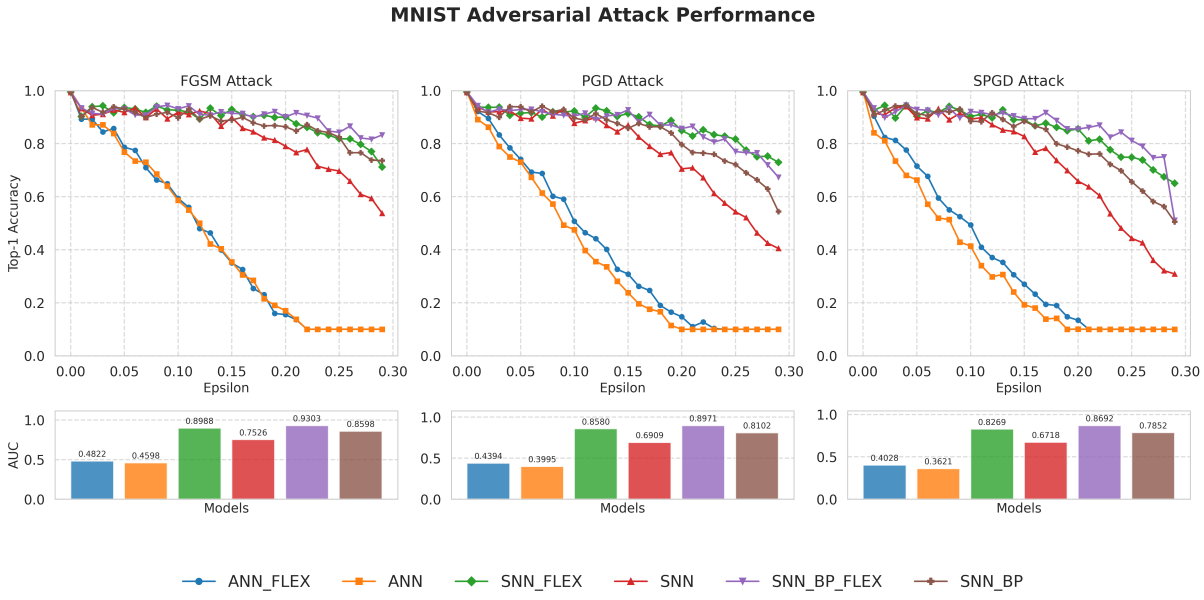


Figure 3.4 MNIST adversarial robustness. Top: accuracy under FGSM, PGD, and SPGD attacks with increasing perturbation ϵ. Bottom: AUC scores summarising robustness across ϵ. Flex-enhanced SNNs consistently achieve the highest robustness.

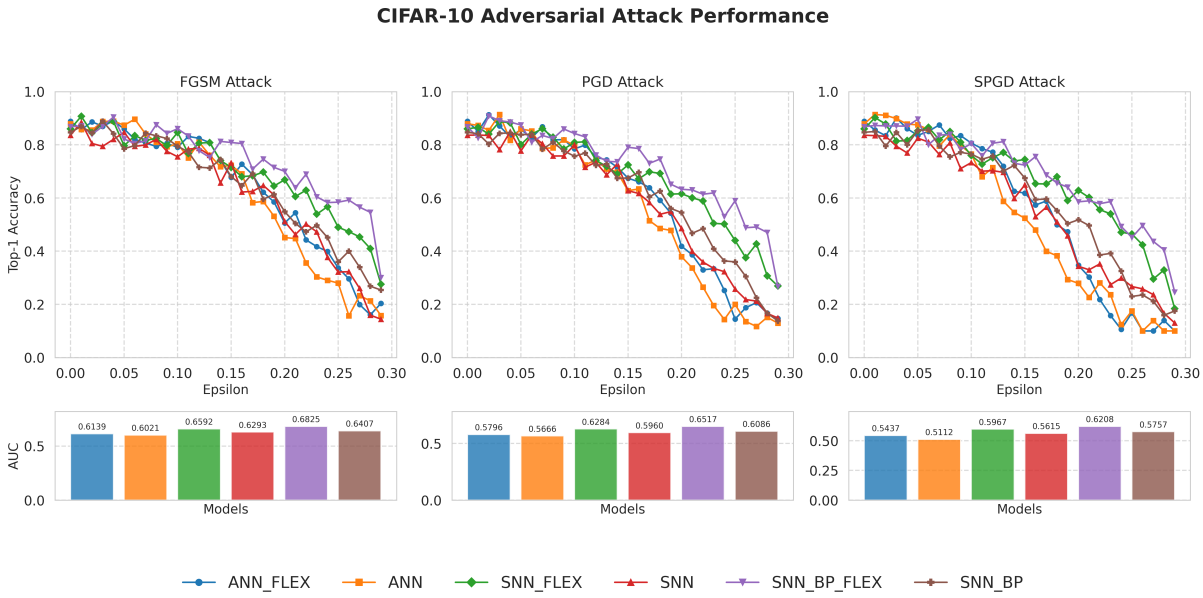


Figure 3.5 CIFAR-10 adversarial robustness.

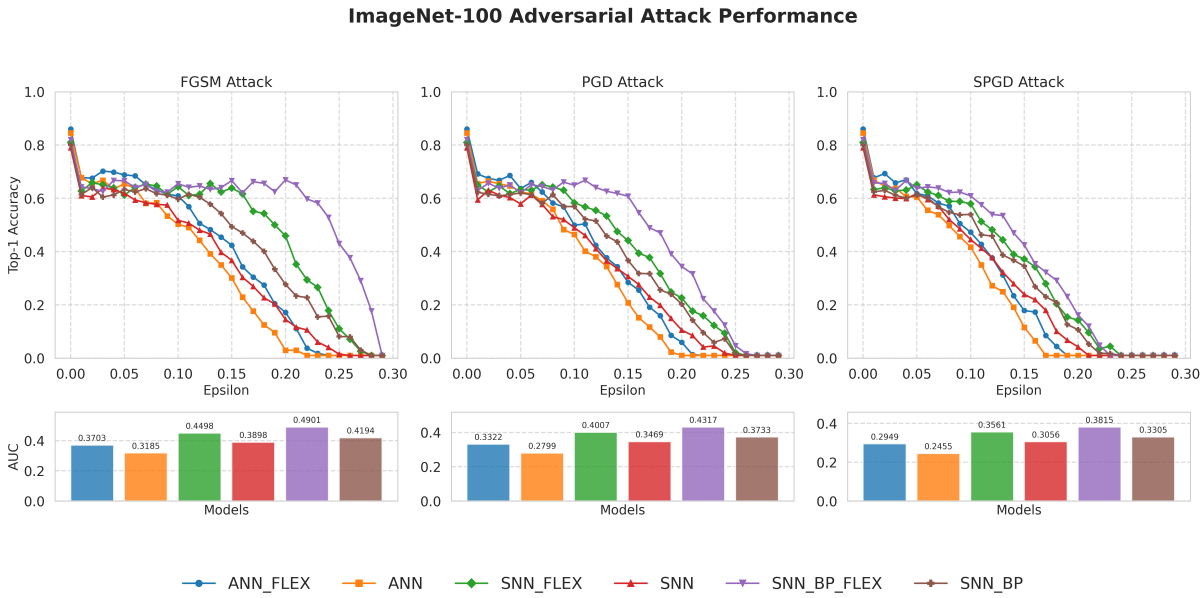
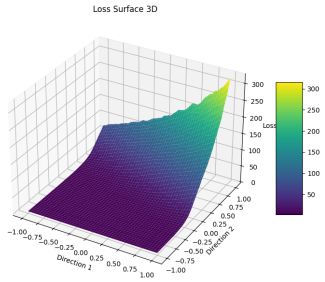


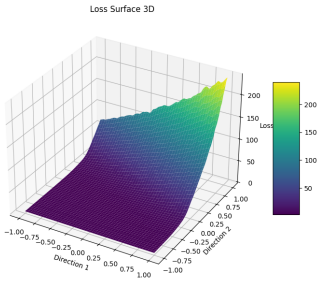
Figure 3.6 ImageNet-100 adversarial robustness.

C.2.2 Loss Landscapes

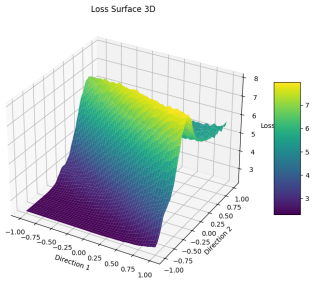
The following figures report the loss landscapes on MNIST, CIFAR-10, and ImageNet-100.



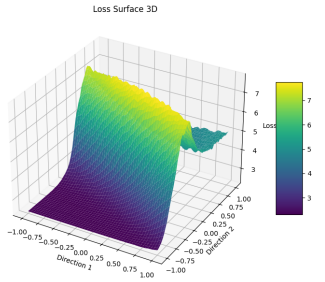
(a) ANN without Flex



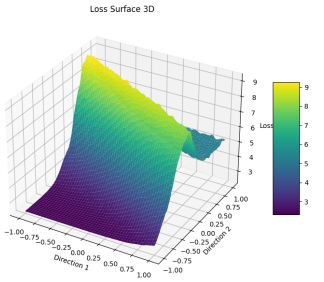
(b) ANN with Flex



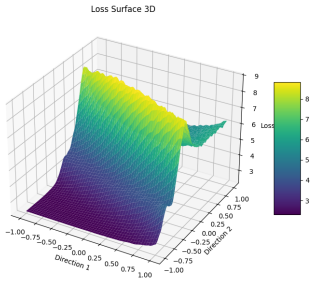
(c) SNN without Flex



(d) SNN with Flex

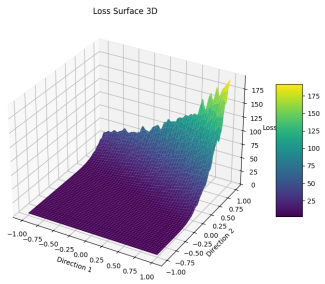


(e) SNN+BP without Flex

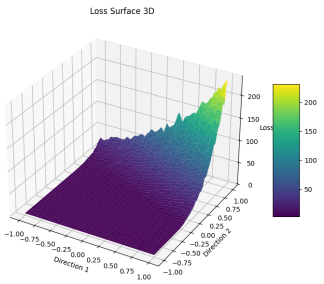


(f) SNN+BP with Flex

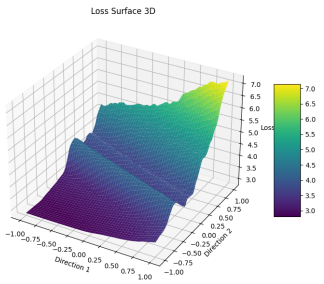
Figure 3.7 Comparison of loss landscapes across six model variants on MNIST dataset.



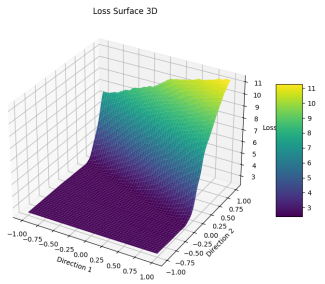
(a) ANN without Flex



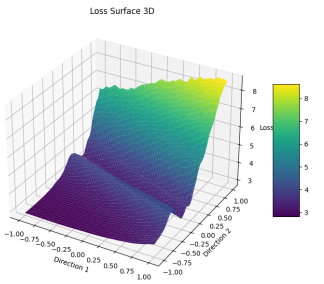
(b) ANN with Flex



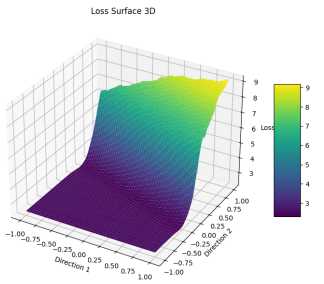
(c) SNN without Flex



(d) SNN with Flex

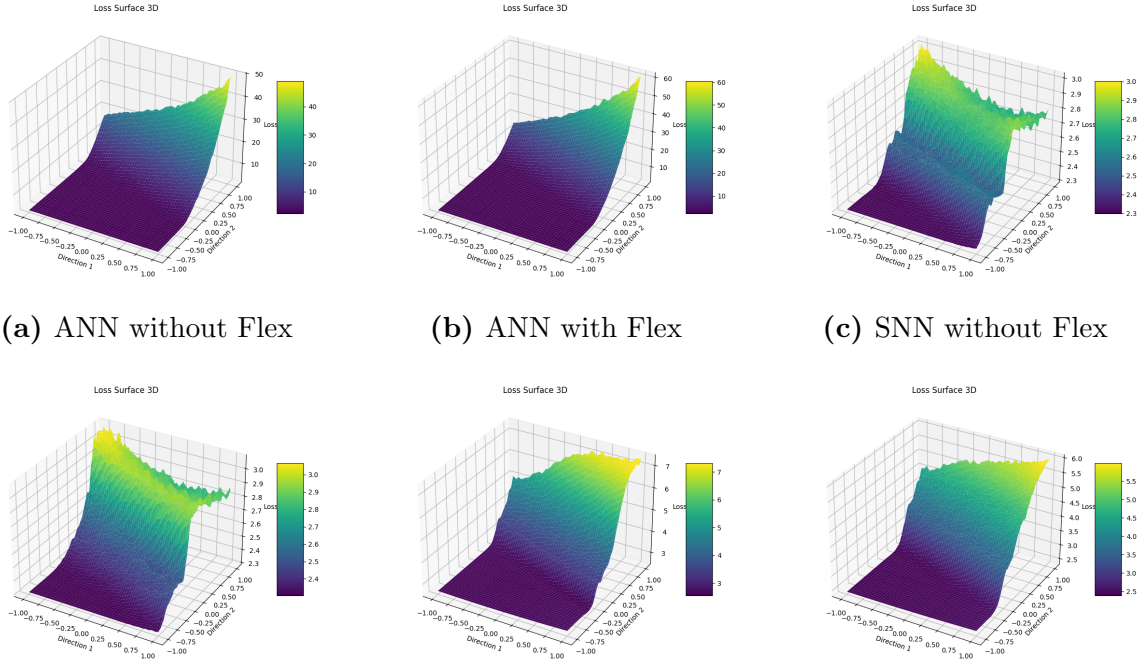


(e) SNN+BP without Flex



(f) SNN+BP with Flex

Figure 3.8 Comparison of loss landscapes across six model variants on CIFAR-10 dataset.



(a) ANN without Flex **(b)** ANN with Flex **(c)** SNN without Flex

(d) SNN with Flex **(e)** SNN+BP without Flex **(f)** SNN+BP with Flex

Figure 3.9 Comparison of loss landscapes across six model variants on ImageNet-100 dataset.

C.3 Additional Interpretability Results

C.3.1 Binariness

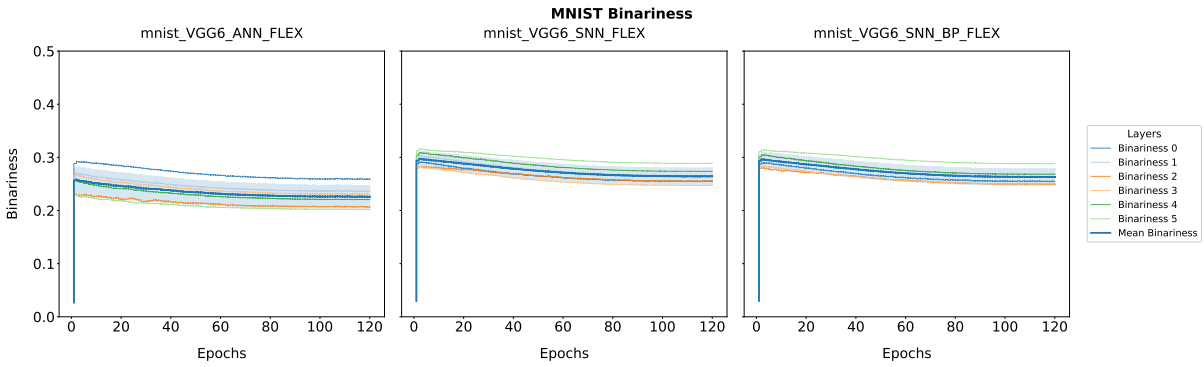


Figure 3.10 Binariness evolution on MNIST across different architectures.

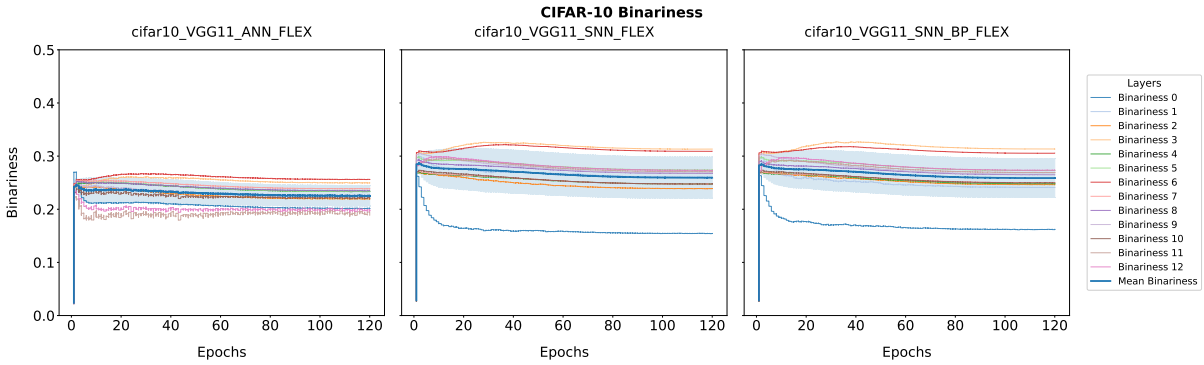


Figure 3.11 Binariness evolution on CIFAR-10 across different architectures.

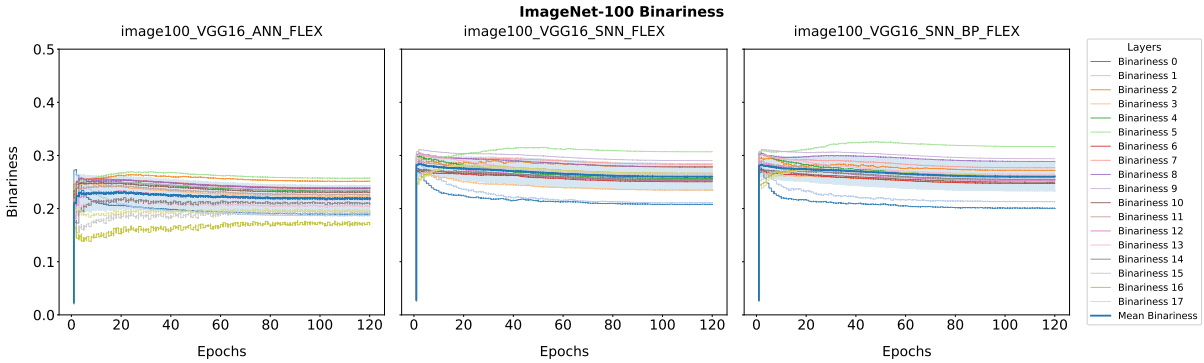


Figure 3.12 Binariness evolution on ImageNet-100 across different architectures.

C.3.2 Conv Ratio–Binariness Evolution and Correlation

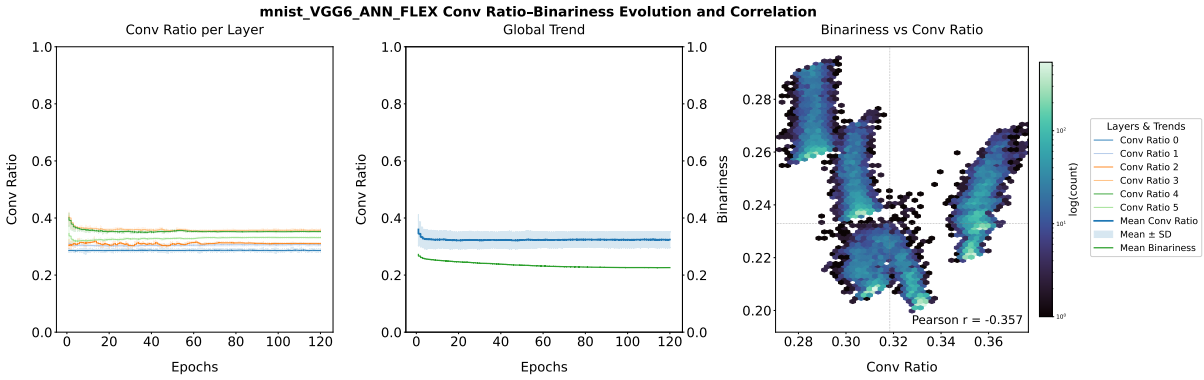


Figure 3.13 MNIST-VGG6-ANN-FLEX Conv Ratio–Binariness Dashboard. The left panel shows the evolution of convolution ratios across layers, the middle panel illustrates the global trend with mean ± SD and mean binariness, and the right panel depicts the correlation between binariness and convolution ratio.

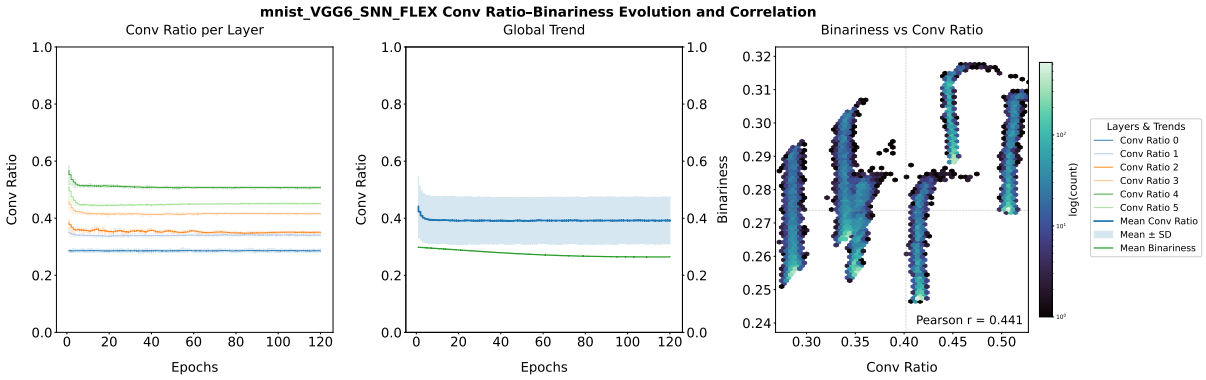


Figure 3.14 MNIST-VGG6-SNN-FLEX Conv Ratio–Binariness Dashboard.

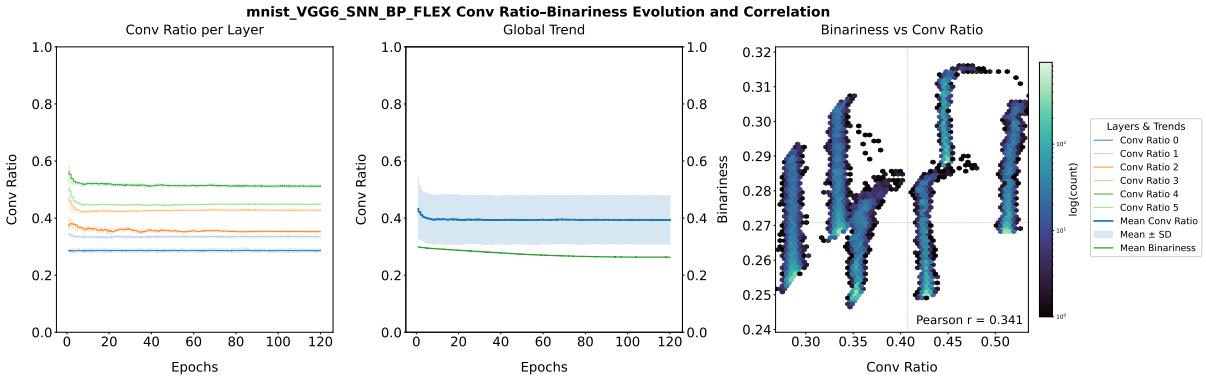


Figure 3.15 MNIST-VGG6-SNN-BP-FLEX Conv Ratio–Binariness Dashboard.

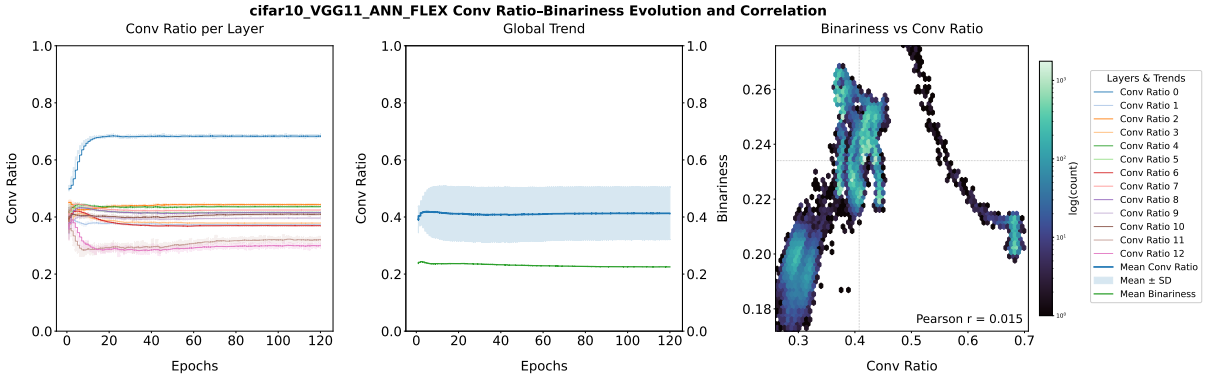


Figure 3.16 CIFAR10-VGG11-ANN-FLEX Conv Ratio–Binariness Dashboard. The left panel shows the evolution of convolution ratios across layers, the middle panel illustrates the global trend with mean ± SD and mean binariness, and the right panel depicts the correlation between binariness and convolution ratio.

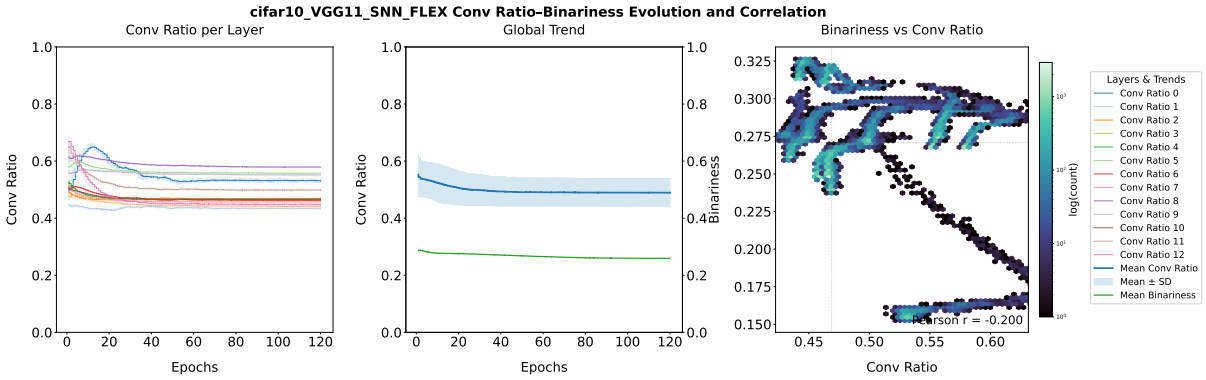


Figure 3.17 CIFAR10-VGG11-SNN-FLEX Conv Ratio–Binariness Dashboard.

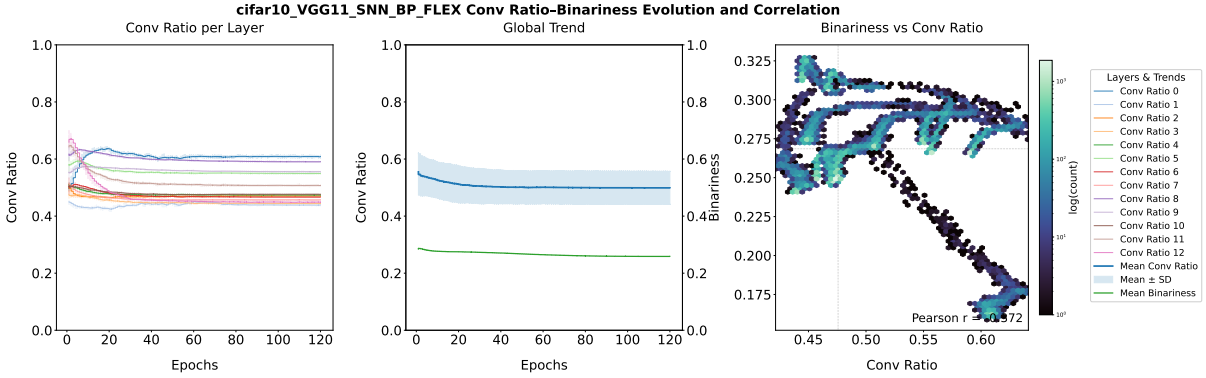


Figure 3.18 CIFAR10-SNN-BP-FLEX Conv Ratio–Binariness Dashboard.

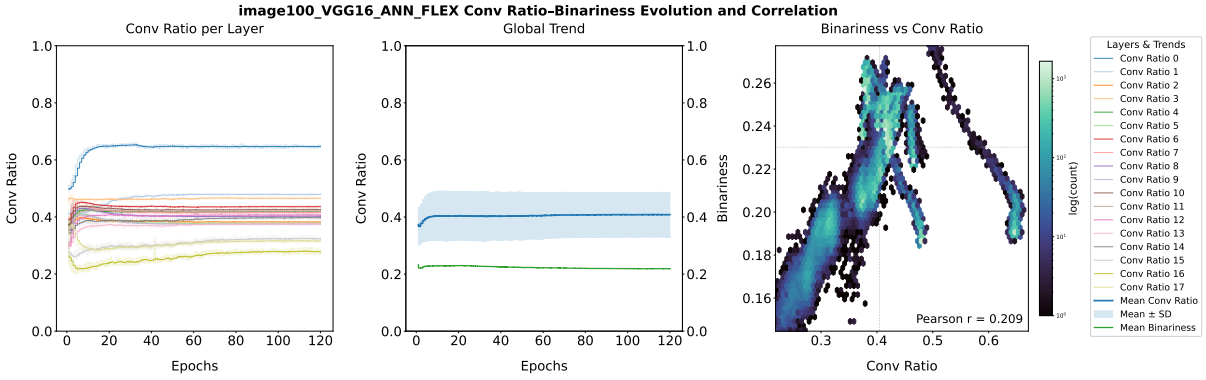


Figure 3.19 ImageNet100-VGG16-ANN-FLEX Conv Ratio–Binariness Dashboard. The left panel shows the evolution of convolution ratios across layers, the middle panel illustrates the global trend with mean ± SD and mean binariness, and the right panel depicts the correlation between binariness and convolution ratio.

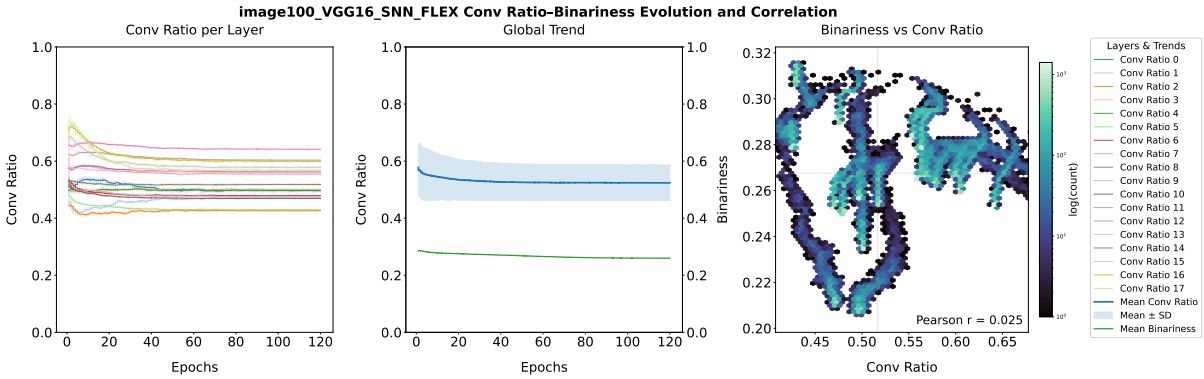


Figure 3.20 ImageNet100-VGG16-SNN-FLEX Conv Ratio–Binariness Dashboard.

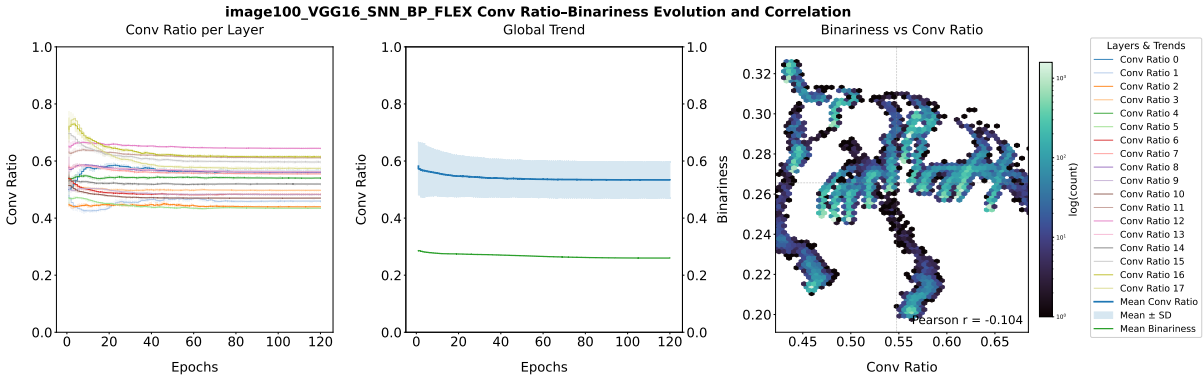


Figure 3.21 ImageNet100-VGG16-SNN-BP-FLEX Conv Ratio–Binariness Dashboard.

C.4 Other Results

Appendix D

Computational Resources and Reproducibility

D.1 Hardware and HPC Platform

All experiments were executed on the Imperial College London High Performance Computing (HPC) facility (Research Computing Service, DOI: 10.14469/hpc/2232). As described in the main text, compute nodes are equipped with dual-socket AMD EPYC 7742 processors (64 cores per socket, 256 hardware threads in total) and 1.0 TiB of system memory, running Linux kernel 4.18.

GPU-accelerated training was performed on NVIDIA Quadro RTX 6000 devices, each with 24 GB GDDR6 memory and CUDA 12.2 support (driver version 535.54.03). Typical jobs requested a single GPU, 2 CPU cores, and 16 GB of system memory, with maximum walltime up to 64 hours. Multi-GPU nodes hosting up to eight RTX 6000 devices were available; memory utilisation per process ranged between 4 GB and 24 GB depending on the experiment.

Jobs were scheduled via the PBS batch system. A representative submission script is shown below:

```
#!/bin/bash
#PBS -l select=1:ncpus=2:mem=16gb:ngpus=1
#PBS -l walltime=64:00:00
#PBS -N analysis
...
python src/train_initialiser.py --config configs/mnist1.json
```

To ensure reproducibility across runs, environment variables limited thread usage (`OMP_NUM_THREADS=1`, `MKL_NUM_THREADS=1`), and the PyTorch CUDA allocator was configured with `expandable_segments=True` to avoid fragmentation. Checkpoints were synchronised between local scratch (`$TMPDIR`) and the project directory at job completion.

D.2 Software Environment

All experiments were run under Red Hat Enterprise Linux 8.5 (Ootpa). The software environment was based on Python 3.12.10, managed with Conda. GPU drivers were provided by the NVIDIA driver version 535.54.03 with CUDA 12.2.

Model training and evaluation were implemented in PyTorch (with torchvision), and relied on standard scientific Python libraries including NumPy, scikit-learn, matplotlib, seaborn, tqdm, and rich. The complete environment configuration is available as a Conda environment file (environment.yml) in the project repository.

D.3 Code Availability

List of References

[1] Nadeem Akhtar and U Ragavendran. "Interpretation of intelligence in CNN-pooling processes: a methodological survey". In: *Neural computing and applications* 32.3 (2020), pp. 879–898.

[2] Ali Almasi et al. "Mechanisms of feature selectivity and invariance in primary visual cortex". In: *Cerebral Cortex* 30.9 (2020), pp. 5067–5087.

[3] Fabio Anselmi, Lorenzo Rosasco, and Tomaso Poggio. "On invariance and selectivity in representation learning". In: *Information and Inference: A Journal of the IMA* 5.2 (2016), pp. 134–158.

[4] Valerio Biscione and Jeffrey S Bowers. "Convolutional neural networks are not invariant to translation, but they can learn to be". In: *Journal of Machine Learning Research* 22.229 (2021), pp. 1–28.

[5] Elvis K Boahen et al. "Bio-Inspired Neuromorphic Sensory Systems from Intelligent Perception to Nervetronics". In: *Advanced Science* 12.1 (2025), p. 2409568.

[6] Charles Cadieu et al. "A model of V4 shape selectivity and invariance". In: *Journal of neurophysiology* 98.3 (2007), pp. 1733–1750.

[7] Shaofeng Cai, Yao Shu, and Wei Wang. "Dynamic routing networks". In: *proceedings of the IEEE/CVF winter conference on applications of computer vision*. 2021, pp. 3588–3597.

[8] Alessia Celeghin et al. "Convolutional neural networks for vision neuroscience: significance, developments, and outstanding issues". In: *Frontiers in Computational Neuroscience* 17 (2023), p. 1153572.

[9] Ahmad Chaddad et al. "Modeling information flow through deep neural networks". In: *arXiv preprint arXiv:1712.00003* (2017).

[10] Yinpeng Chen et al. "Dynamic convolution: Attention over convolution kernels". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 11030–11039.

[11] Leon O Chua. "CNN: A vision of complexity". In: *International Journal of Bifurcation and Chaos* 7.10 (1997), pp. 2219–2425.

[12] Jenna M Crowe-Riddell and Harvey B Lillywhite. "Sensory systems". In: *Health and welfare of captive reptiles*. Springer, 2023, pp. 45–91.

[13] Prachi Garg et al. *Memorization and generalization in deep cnns using soft gating mechanisms*. 2019.

[14] Stuart Geman. "Invariance and selectivity in the ventral visual pathway". In: *Journal of Physiology-Paris* 100.4 (2006), pp. 212–224.

[15] Ziv Goldfeld et al. "Estimating information flow in deep neural networks". In: *arXiv preprint arXiv:1810.05728* (2018).

[16] Yu He et al. "A trust update mechanism based on reinforcement learning in underwater acoustic sensor networks". In: *IEEE Transactions on Mobile Computing* 21.3 (2020), pp. 811–821.

[17] Yuhang Li et al. "Error-aware conversion from ANN to SNN via post-training parameter calibration". In: *International Journal of Computer Vision* 132.9 (2024), pp. 3586–3609.

[18] Grace W Lindsay. "Convolutional neural networks as a model of the visual system: Past, present, and future". In: *Journal of cognitive neuroscience* 33.10 (2021), pp. 2017–2031.

[19] Yuling Luo et al. "An efficient, low-cost routing architecture for spiking neural network hardware implementations". In: *Neural Processing Letters* 48.3 (2018), pp. 1777–1788.

[20] Quoc Trung Pham et al. "A review of SNN implementation on FPGA". In: *2021 international conference on multimedia analysis and pattern recognition (MAPR)*. IEEE. 2021, pp. 1–6.

[21] Ivet Rafegas et al. "Understanding trained CNNs by indexing neuron selectivity". In: *Pattern Recognition Letters* 136 (2020), pp. 318–325.

[22] Tamás Roska et al. "The use of CNN models in the subcortical visual pathway". In: *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications* 40.3 (1993), pp. 182–195.

[23] Benedetta Savelli et al. "A multi-context CNN ensemble for small lesion detection". In: *Artificial intelligence in medicine* 103 (2020), p. 101749.

[24] Tatyana O Sharpee. "How invariant feature selectivity is achieved in cortex". In: *Frontiers in synaptic neuroscience* 8 (2016), p. 26.

[25] Jinming Su et al. "Selectivity or invariance: Boundary-aware salient object detection". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 3799–3808.

[26] Hoyoung Tang et al. “Spike counts based low complexity SNN architecture with binary synapse”. In: *IEEE Transactions on Biomedical Circuits and Systems* 13.6 (2019), pp. 1664–1677.

[27] Chunwei Tian et al. “Attention-guided CNN for image denoising”. In: *Neural Networks* 124 (2020), pp. 117–129.

[28] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[29] Beilun Wang et al. “How decisions are made in brains: Unpack “black box” of CNN with Ms. Pac-Man video game”. In: *IEEE Access* 8 (2020), pp. 142446–142458.

[30] Xingxing Wei and Shiji Zhao. “Boosting adversarial transferability with learnable patch-wise masks”. In: *IEEE Transactions on Multimedia* 26 (2023), pp. 3778–3787.

[31] Fu Xing et al. “Homeostasis-based cnn-to-snn conversion of inception and residual architectures”. In: *International Conference on Neural Information Processing*. Springer. 2019, pp. 173–184.

[32] Derek Xu et al. “Neural Network Pruning for Invariance Learning”. In: *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*. 2025, pp. 1691–1702.

[33] Shuangming Yang et al. “Self-supervised high-order information bottleneck learning of spiking neural network for robust event-based optical flow estimation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024).

[34] Penghang Yin et al. “Understanding straight-through estimator in training activation quantized neural nets”. In: *arXiv preprint arXiv:1903.05662* (2019).

[35] Shujian Yu et al. “Understanding convolutional neural networks with information theory: An initial exploration”. In: *IEEE transactions on neural networks and learning systems* 32.1 (2020), pp. 435–442.

[36] Jingwei Zhang, Tongliang Liu, and Dacheng Tao. “An information-theoretic view for deep learning”. In: *arXiv preprint arXiv:1804.09060* (2018).

[37] Wenrui Zhang and Peng Li. “Information-theoretic intrinsic plasticity for online unsupervised learning in spiking neural networks”. In: *Frontiers in neuroscience* 13 (2019), p. 31.

[38] Ruohong Zhou. “A method of converting ann to snn for image classification”. In: *2023 IEEE 3rd International Conference on Electronic Technology, Communication and Information (ICETCI)*. IEEE. 2023, pp. 819–822.