Deep Learning Assignment 1 Report M23CSA510

Summary

By using Convolutional Neural Network (CNN) models to learn to classify the MNIST and Caltech-101 datasets in the assignment, I evaluated the impact of different pooling techniques, optimisers, and model topologies on performance. This was done to acquire a more in-depth understanding of how these parameters influence model accuracy and learning, particularly when dealing with less complex datasets, such as MNIST, as opposed to more complex datasets, such as Caltech-101.

Overview of Datasets

1. MNIST Dataset:

MNIST dataset is a well-established benchmark for image classification tasks, especially for handwritten digit recognition. The dataset consists of 28x28 grayscale images of numbers. Since CNNs often perform better with larger input sizes, I resized the images to 128x128 to allow the model to capture more spatial information.

2. Caltech-101 Dataset:

The Caltech-101 dataset has many images of 101 distinct objects. The dataset has varying images of various sizes. I resized the images to 224x224 to ensure uniformity and used RGB, i.e. three channels, to allow the model to leverage colour information. For this dataset, I designed two models, one standard and one deep architecture, and different pooling techniques to understand the impact of depth on accuracy.

Architectures of Models

1. MNIST Model:

I built a basic CNN consisting of two convolutional layers with one pooling layer in between followed by fully connected layers.

This model was suitable for MNIST while testing the impact of different pooling strategies and optimisers on performance.

The two convolutional layers incrementally extracted features from the images, followed by a pooling layer to reduce spatial dimensions. Pooling helps reduce computational complexity and capture critical features. Fully connected layers were used to map the feature space into class probabilities.

2. Caltech-101 Model:

The Caltech101Model one was similar to the MNIST model, with two convolutional layers and one pooling layer followed by fully connected layers. I experimented with both Max Pooling and Average Pooling techniques.

3. Enhanced Caltech-101 Model:

To achieve better results, I built a deeper model with five convolutional layers, pooling after each layer, batch normalisation, and dropout layers.

A deeper model was needed to capture more features of the Caltech-101 dataset. Increasing the number of convolutional layers allowed the model to learn more refined, abstract features, improving its ability to categorise various objects across 101 distinct classes.

The deeper architecture, batch normalisation, and dropout improved the model's generalisation abilities. Batch normalisation stabilised training by normalising activations, while dropout reduced overfitting by randomly ignoring some neurons during training.

Metrics and Evaluation Parameters

1. Loss Function (CrossEntropyLoss):

CrossEntropyLoss was used to measure the difference between the predicted probability and actual class distribution.

- Why: CrossEntropyLoss is effective for classification problems because it penalises incorrect predictions probabilistically, helping the model learn by minimising this difference.

2. Top-1 Accuracy:

This metric measures whether the class with the highest predicted probability matches the actual class.

- Why: Top-1 accuracy is crucial for assessing the model's best prediction, especially in MNIST, where the model needs to predict a single digit out of 10.

3. Top-5 Accuracy:

This metric measures whether the class is within the top five predicted classes.

- Why: Top-5 accuracy benefits the Caltech-101 dataset, where the model has to choose between 101 different categories. It helps assess whether the model is "close" to the correct prediction, even if it doesn't always choose the top one.

4. Training and Validation Loss:

I tracked training and validation losses to see how well the model learned from the training data and how well it generalises to unseen data.

- Why: Monitoring these losses helped in detecting overfitting. If the training loss decreased but the validation loss stayed high, the model was overfitting to the training data.

Training Methodology

1. Data Augmentation:

I applied data augmentation techniques, including random rotations, colour jittering, and horizontal flips, to the Caltech-101 dataset.

- Why: Data augmentation helps the model generalise better by increasing the diversity of the training data. This is especially useful for datasets with limited samples and diverse categories.

2. Pooling Strategies:

I experimented with Max Pooling and Average Pooling in my models.

- Why: Pooling reduces the spatial dimensions of feature maps, which helps reduce computation and avoid overfitting. Max Pooling captures the most significant features, while Average Pooling smooths out the feature map.

3. Optimizers:

I used Adam and SGD optimisers in different setups. Adam adapts the learning rate dynamically, while SGD uses a fixed learning rate with momentum.

- Why: Adam provides faster convergence, making it ideal for more straightforward datasets like MNIST. However, for deeper architectures like the one used for Caltech-101, SGD with momentum provided more stable long-term performance.

4. Learning Rate Scheduler:

I used a cosine annealing learning rate scheduler for the Caltech-101 dataset.

- Why: This scheduler gradually reduces the learning rate, helping the model make more minor, more refined updates during the later stages of training, ensuring convergence without overshooting.

5. Mixed Precision Training:

I used mixed precision training for the deeper Caltech-101 model to reduce memory usage and speed up computations.

- Why: Mixed precision training allows faster training without sacrificing accuracy, which is particularly useful for deep models that require significant computational resources.

Model Results

MNIST Results

- Top-1 and Top-5 Accuracy: Nearly 99% Top-1, 100% Top-5 accuracy was achieved in both models with Max and Average Pooling along with Adam optimiser; SGD fell behind by a few percentage points to around97% for Top-1 but 100% for Top-5.

- Training Time: The models converged within ten epochs, each taking approximately 30-60 seconds.

- Loss: The training loss dropped to near zero in early epochs, while validation loss closely tracked the training loss, indicating strong generalisation.

Max Pooling slightly outperformed Average Pooling by focusing on the most significant features, which works well for more straightforward datasets like MNIST.

Adam converged faster than SGD, making it ideal for a straightforward classification task like MNIST.

Caltech-101 Results

- Top-1 Accuracy and Top-5 Accuracy: The best-performing setup using SGD with Max Pooling achieved around 51% Top-1 accuracy and 68% Top-5 accuracy. Adam's Optimizer could not perform as well as SGD, with the best performance with Max pooling, which hovered around 40% for Top-1 and 57% for Top-5.

- Training Time: Each epoch took about 1-2 minutes, and the model required around 30 epochs to achieve stable results.

- Loss: Validation loss was consistently lower with Max Pooling, showing that it helped capture key features better than Average Pooling.

Adam converged faster initially, but as the models became more complex, SGD provided a more stable training process, especially when combined with the cosine annealing learning rate scheduler.

Caltech-101 Enhanced Model Results

- Top-1 Accuracy: The enhanced model using Max Pooling and SGD reached 66% Top-1 accuracy and 85% Top-5 accuracy after 30 epochs. I enhanced the model as well. The Adam optimiser lagged with the best results, with Max pooling at Top-1 of 41% and Top-5 of 58%

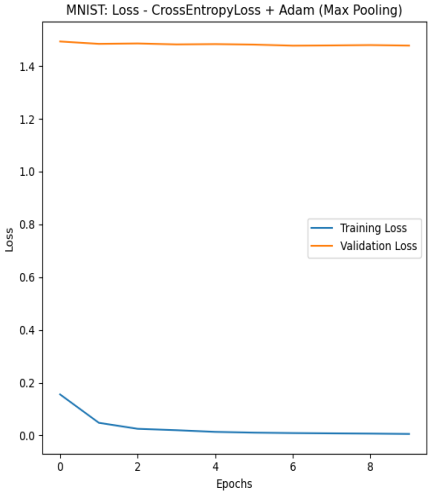
- Training Time: Each epoch took approximately 1-3 minutes due to the increased complexity of the model.

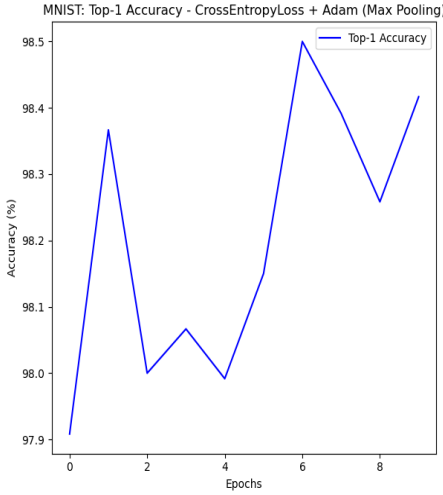
- Loss: Both training and validation losses showed a consistent decrease, indicating that the deeper model generalised better than the baseline models.

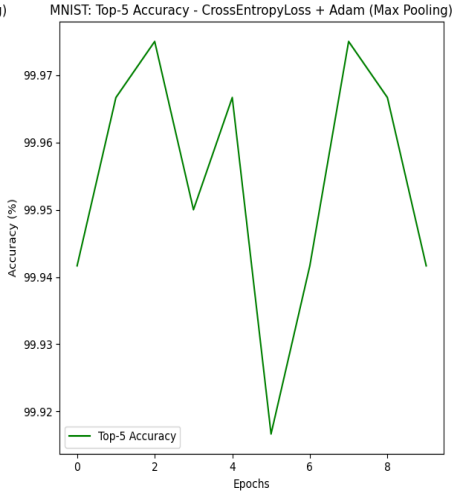
The above results showed that increasing the depth allowed the model to learn more abstract features, which is necessary for handling complex datasets like Caltech-101. Dropout reduced overfitting by forcing the model to rely on a broader range of features, while batch normalisation stabilised training, allowing for faster and more effective learning.

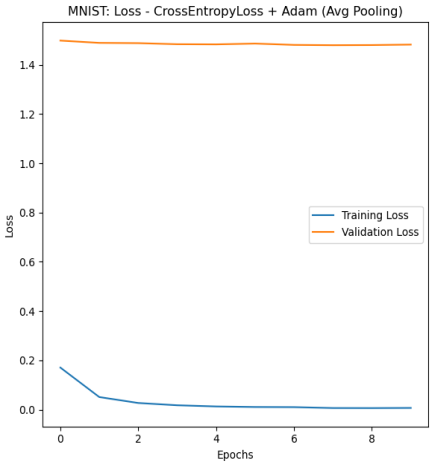
Loss per epoch and accuracy per epoch plots

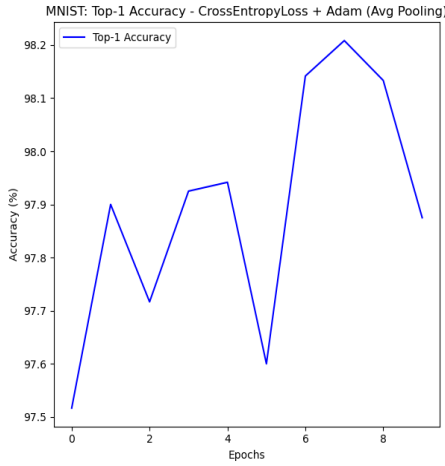
MNIST

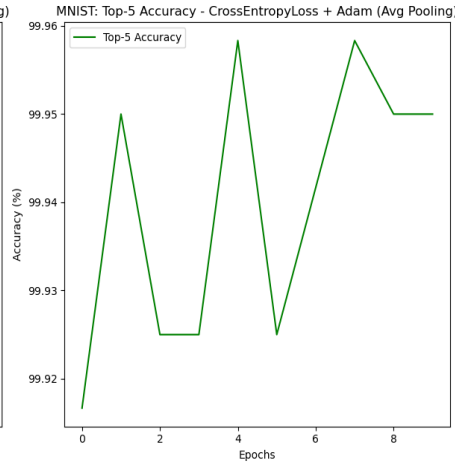


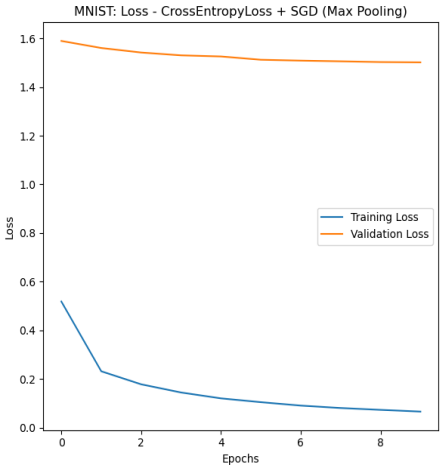


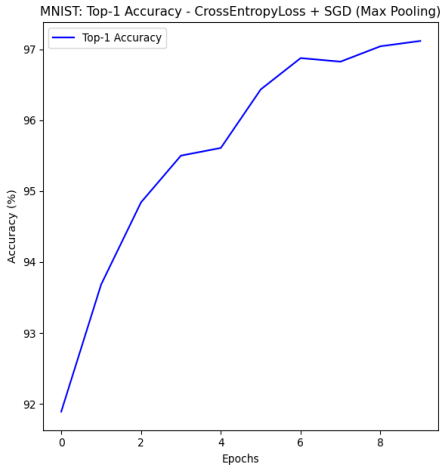


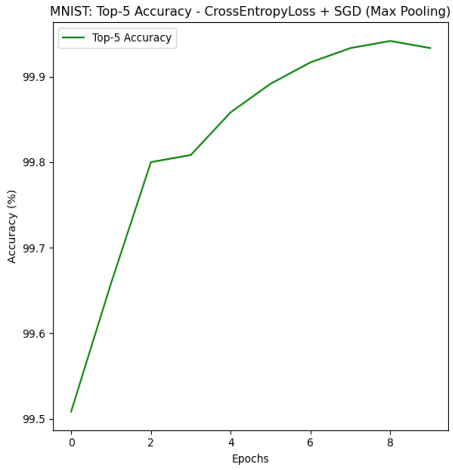


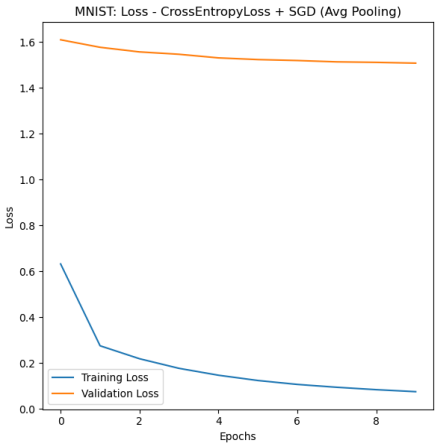


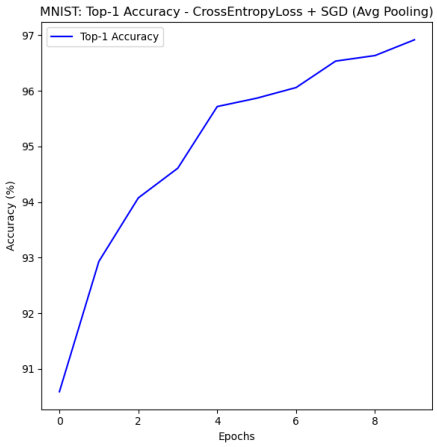


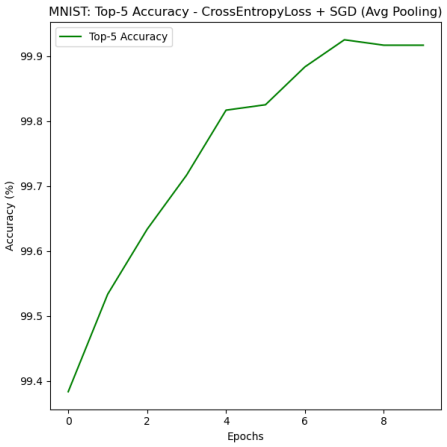




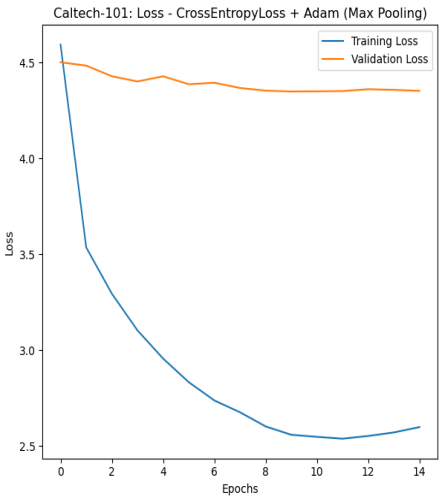


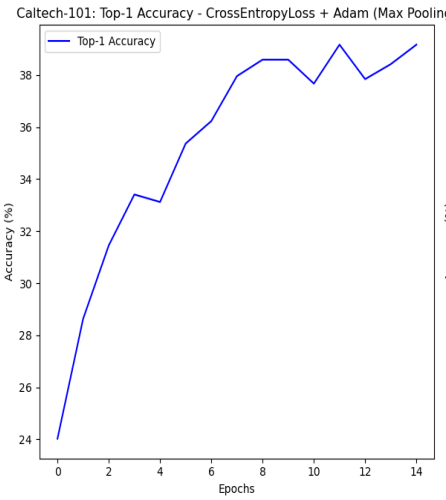

MNIST: Loss - CrossEntropyLoss + SGD (Avg Pooling)
Training Loss
Validation Loss
Loss
Epochs

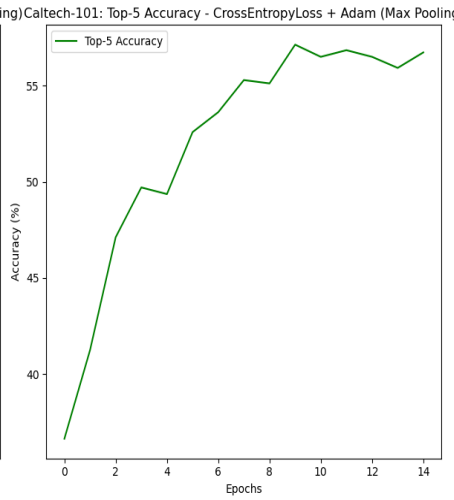

MNIST: Top-1 Accuracy - CrossEntropyLoss + SGD (Avg Pooling)
Top-1 Accuracy
Accuracy (%)
Epochs

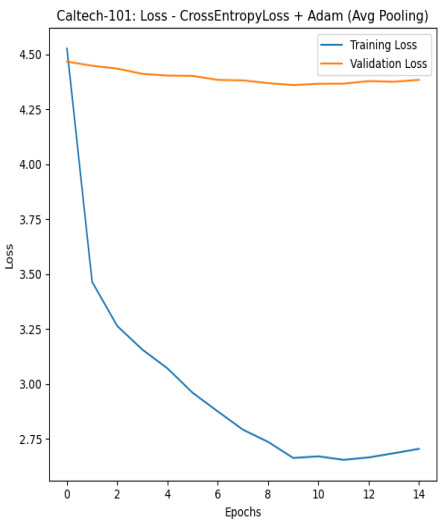

MNIST: Top-5 Accuracy - CrossEntropyLoss + SGD (Avg Pooling)
Top-5 Accuracy
Accuracy (%)
Epochs

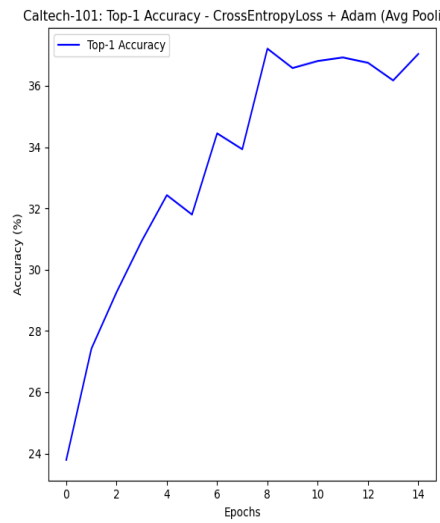
Caltech 101

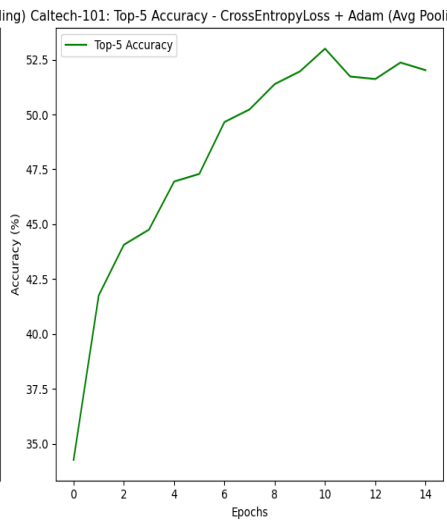

Caltech-101: Loss - CrossEntropyLoss + Adam (Max Pooling)
Training Loss
Validation Loss
Loss
Epochs

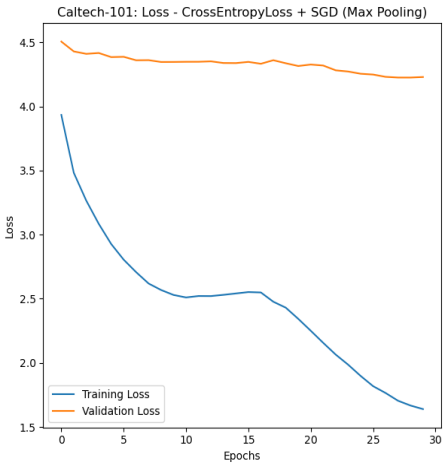

Caltech-101: Top-1 Accuracy - CrossEntropyLoss + Adam (Max Pooling)
Top-1 Accuracy
Accuracy (%)
Epochs

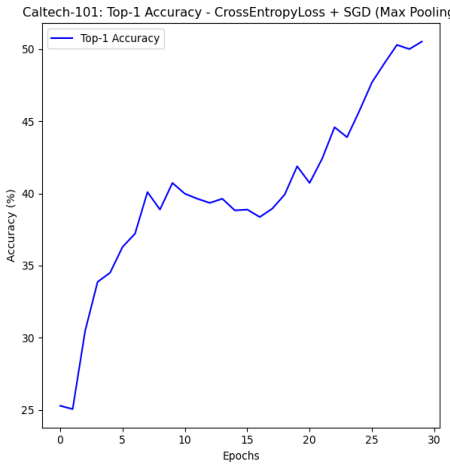

Caltech-101: Top-5 Accuracy - CrossEntropyLoss + Adam (Max Pooling)
Top-5 Accuracy
Accuracy (%)
Epochs

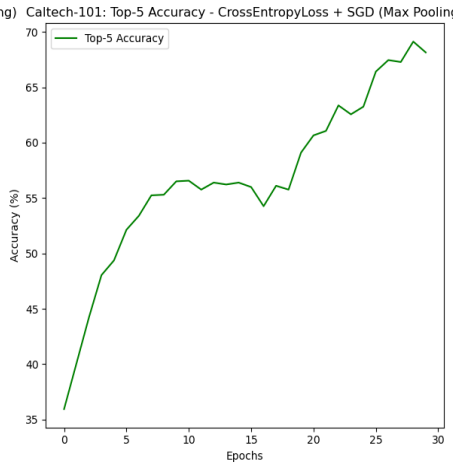

Caltech-101: Loss - CrossEntropyLoss + Adam (Avg Pooling)
Training Loss
Validation Loss
Loss
Epochs

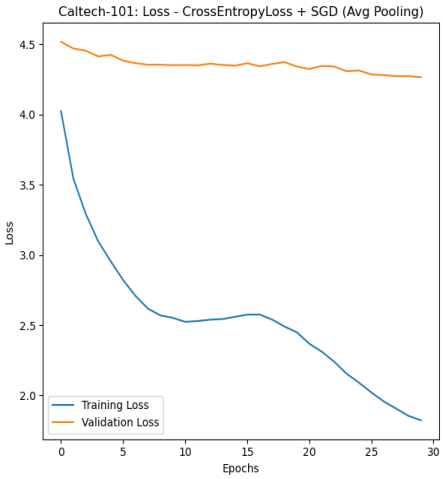

Caltech-101: Top-1 Accuracy - CrossEntropyLoss + Adam (Avg Pooling)
Top-1 Accuracy
Accuracy (%)
Epochs

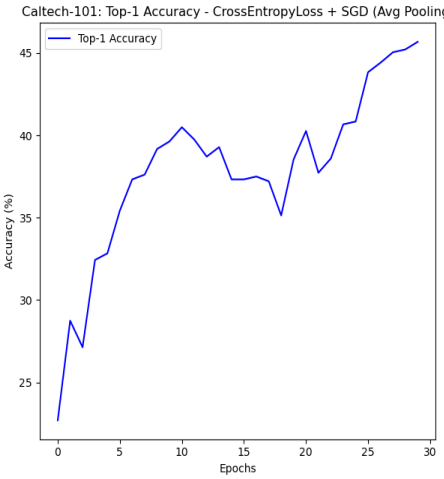

Caltech-101: Top-5 Accuracy - CrossEntropyLoss + Adam (Avg Pooling)
Top-5 Accuracy
Accuracy (%)
Epochs

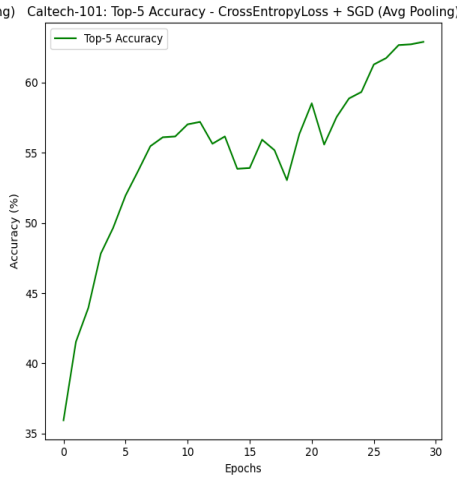




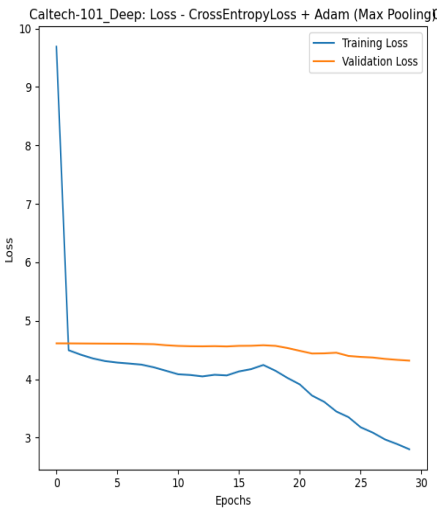


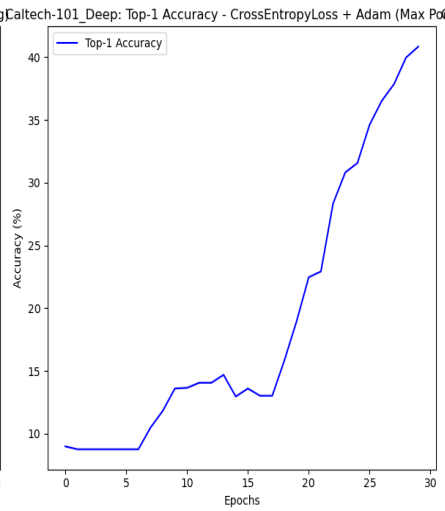


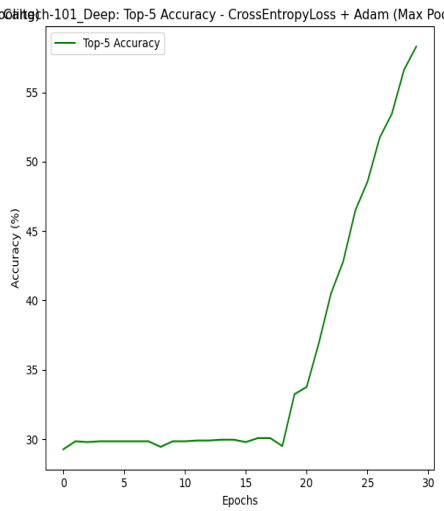


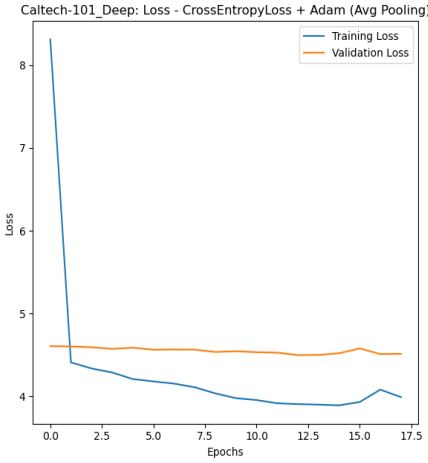


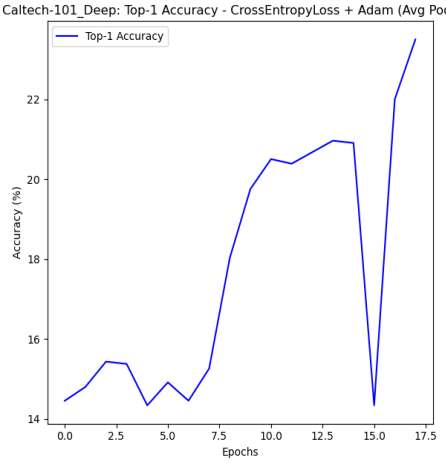
Caltech 101 Deep

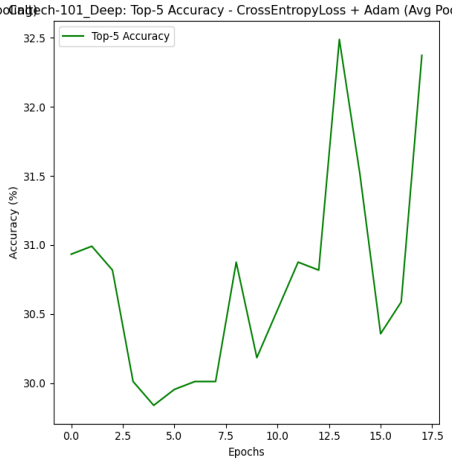


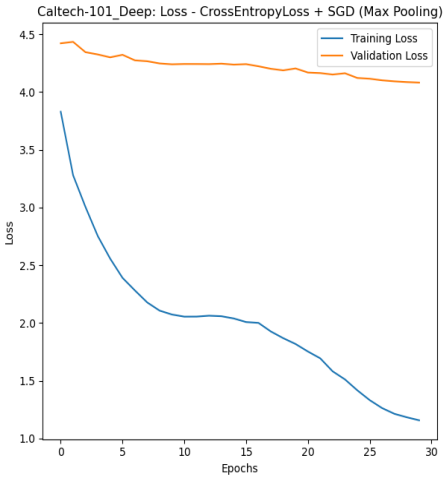


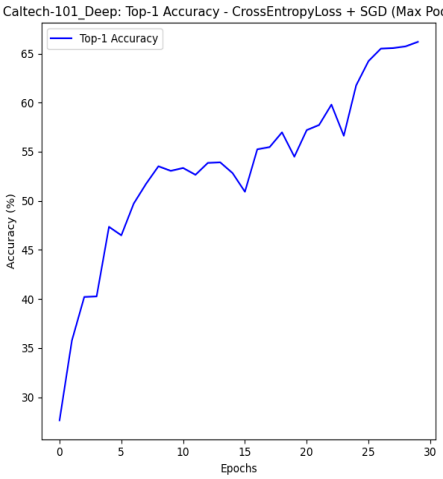


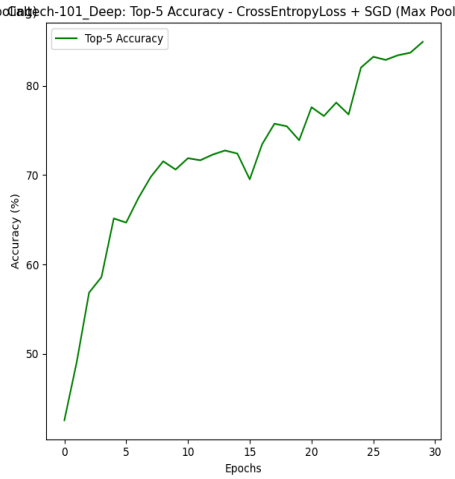

Caltech-101_Deep: Loss - CrossEntropyLoss + Adam (Avg Pooling)
Training Loss
Validation Loss
Loss
Epochs

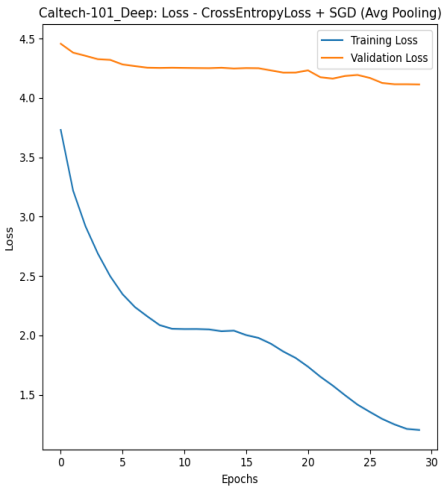

Caltech-101_Deep: Top-1 Accuracy - CrossEntropyLoss + Adam (Avg Pooling)
Top-1 Accuracy
Accuracy (%)
Epochs

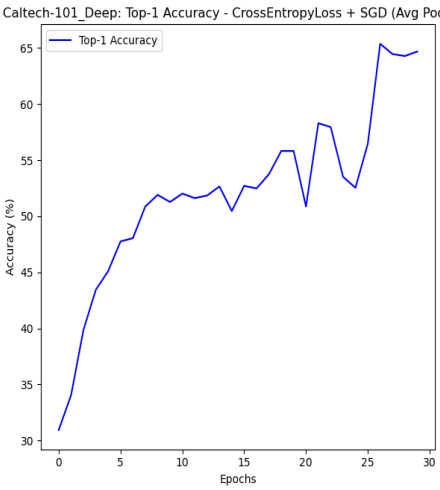

Caltech-101_Deep: Top-5 Accuracy - CrossEntropyLoss + Adam (Avg Pooling)
Top-5 Accuracy
Accuracy (%)
Epochs

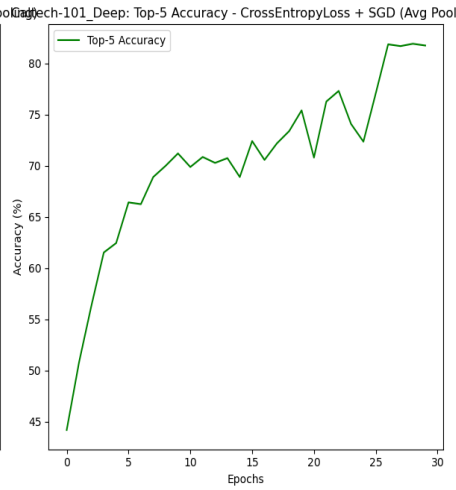

Caltech-101_Deep: Loss - CrossEntropyLoss + SGD (Max Pooling)
Training Loss
Validation Loss
Loss
Epochs


Caltech-101_Deep: Top-1 Accuracy - CrossEntropyLoss + SGD (Max Pooling)
Top-1 Accuracy
Accuracy (%)
Epochs

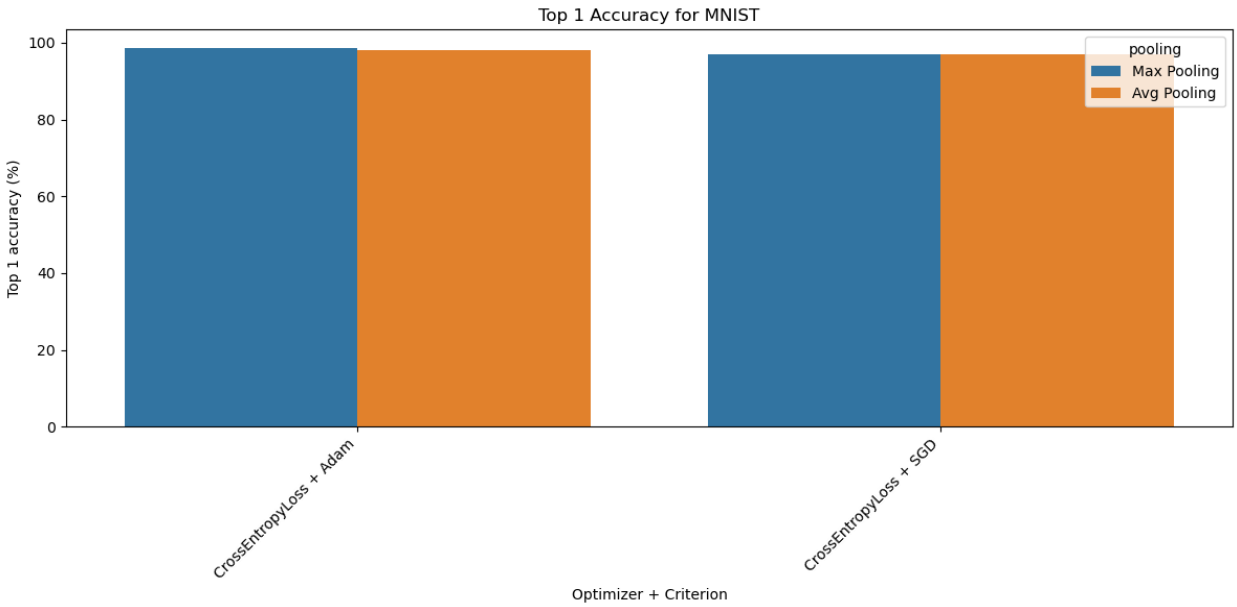

Caltech-101_Deep: Top-5 Accuracy - CrossEntropyLoss + SGD (Max Pooling)
Top-5 Accuracy
Accuracy (%)
Epochs

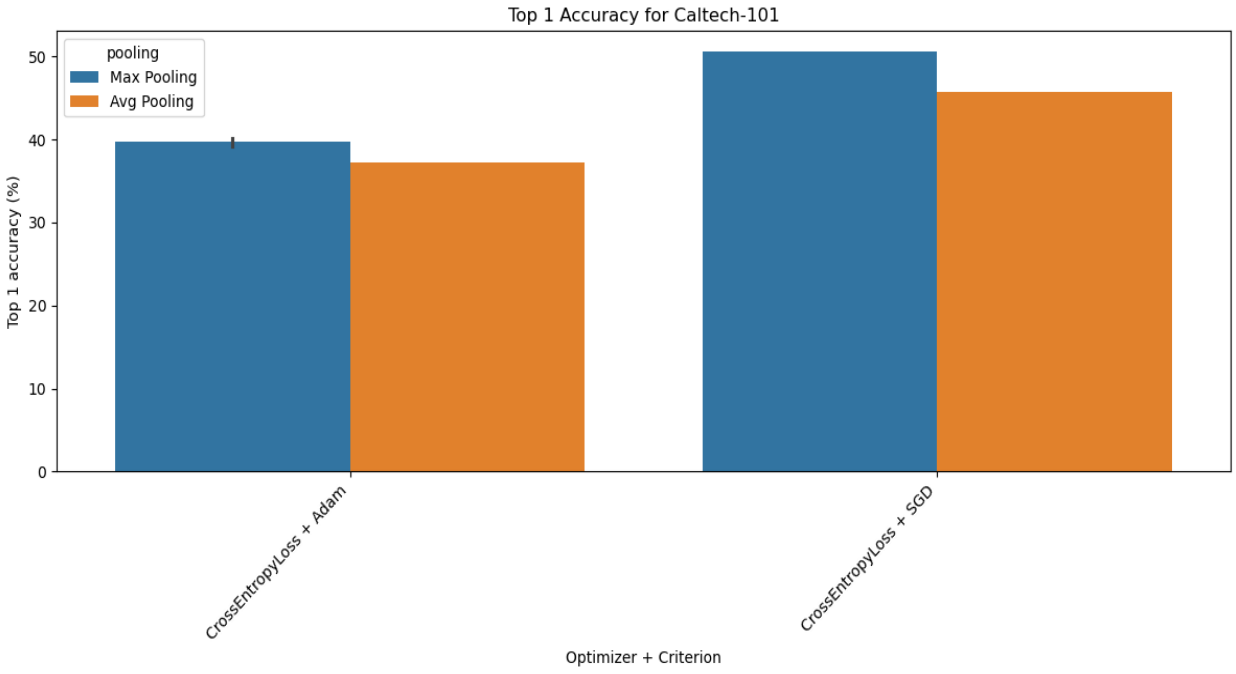

Caltech-101_Deep: Loss - CrossEntropyLoss + SGD (Avg Pooling)
Training Loss
Validation Loss
Loss
Epochs

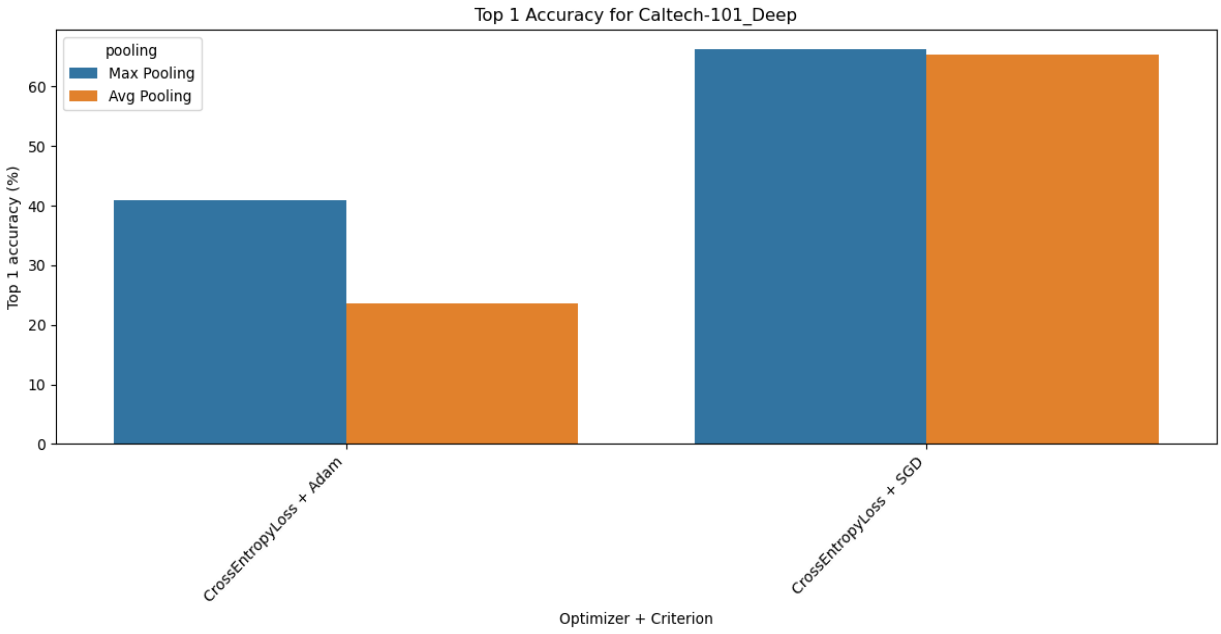

Caltech-101_Deep: Top-1 Accuracy - CrossEntropyLoss + SGD (Avg Pooling)
Top-1 Accuracy
Accuracy (%)
Epochs


Caltech-101_Deep: Top-5 Accuracy - CrossEntropyLoss + SGD (Avg Pooling)
Top-5 Accuracy
Accuracy (%)
Epochs

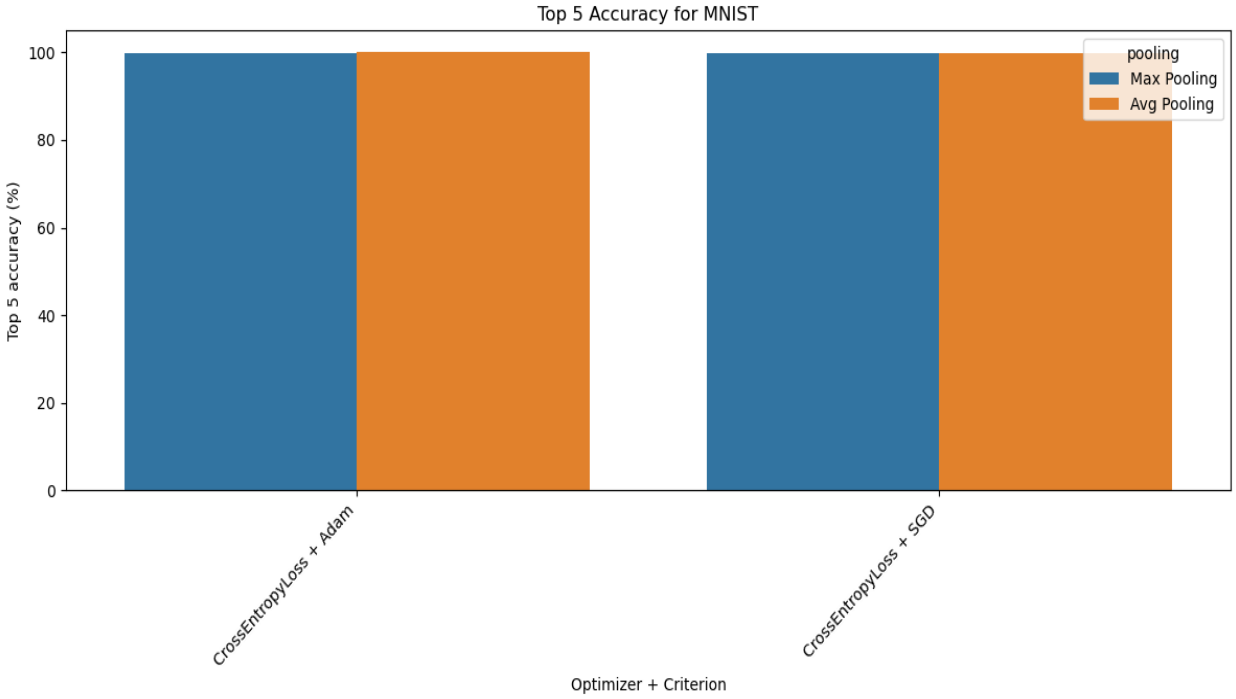
Top-1 Accuracy plots

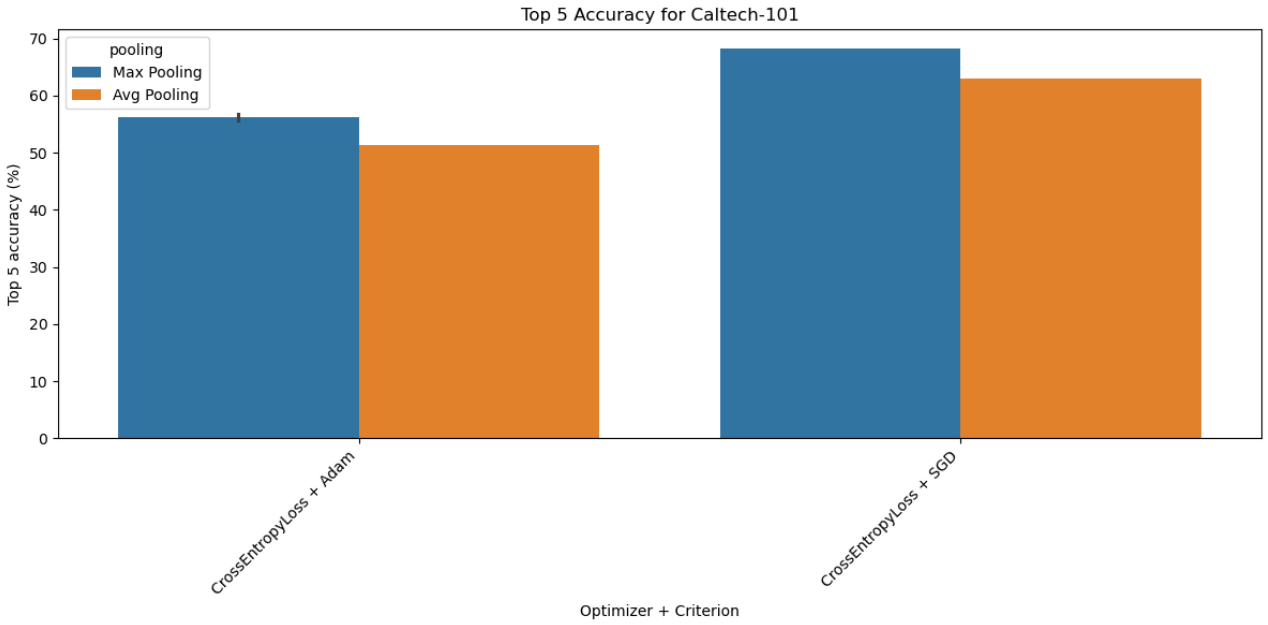


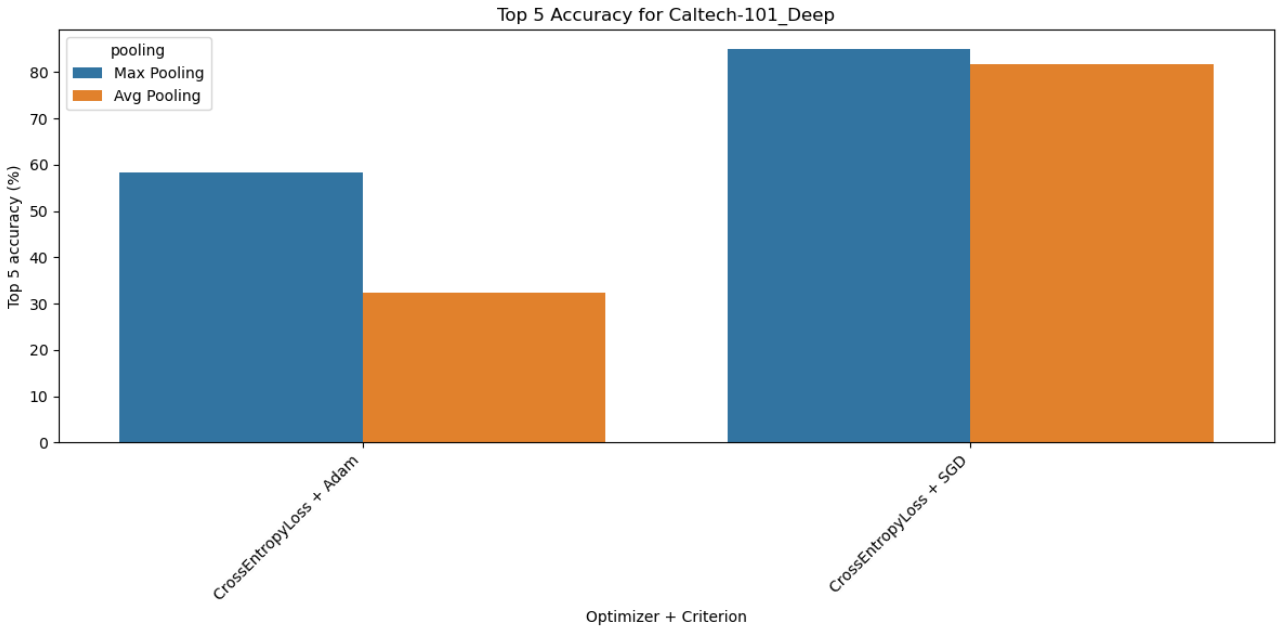




Top-5 Accuracy plots

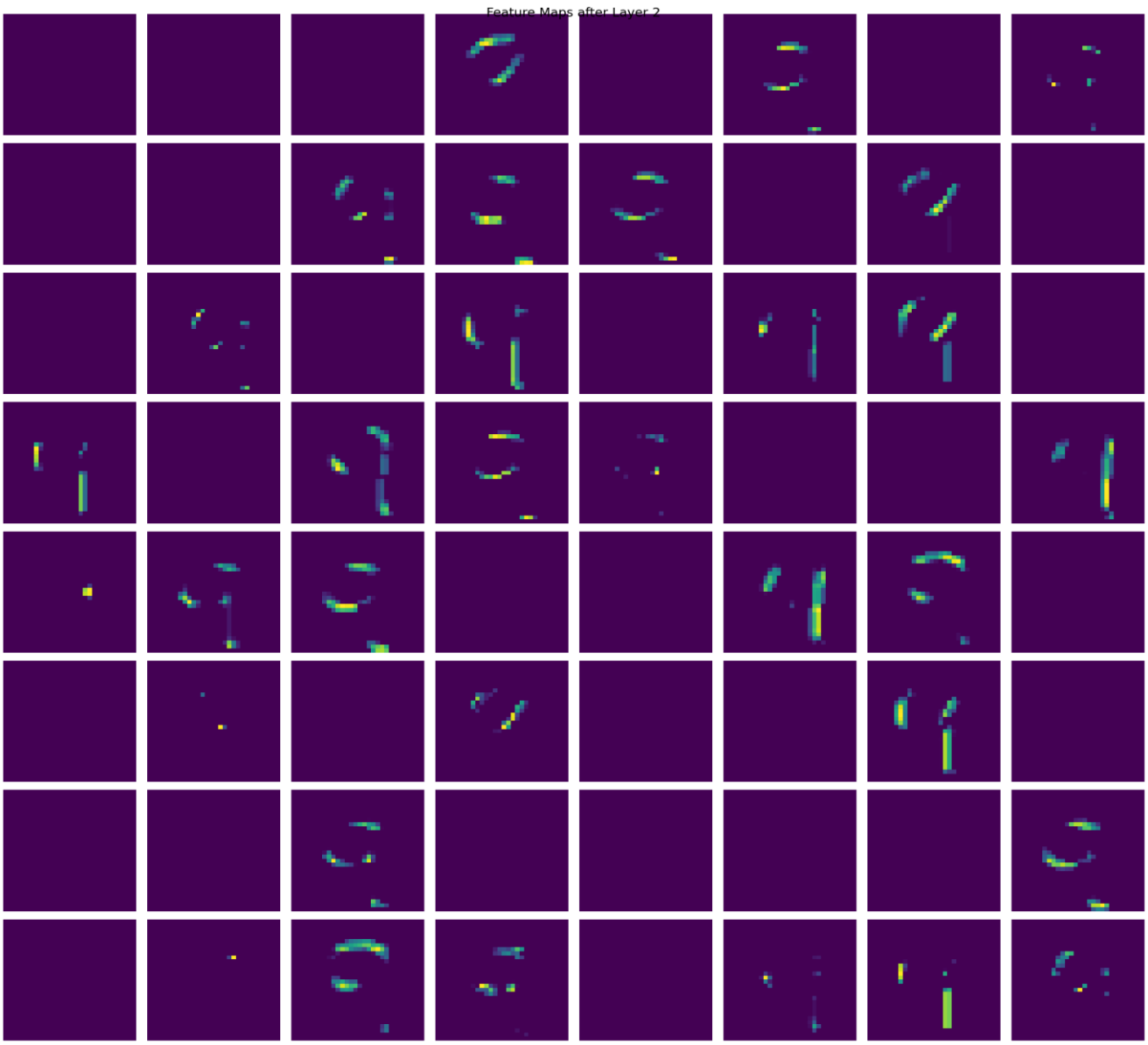



Top 5 Accuracy for Caltech-101
pooling
Max Pooling
Avg Pooling
Top 5 accuracy (%)
0
10
20
30
40
50
60
70
CrossEntropyLoss + Adam
CrossEntropyLoss + SGD
Optimizer + Criterion

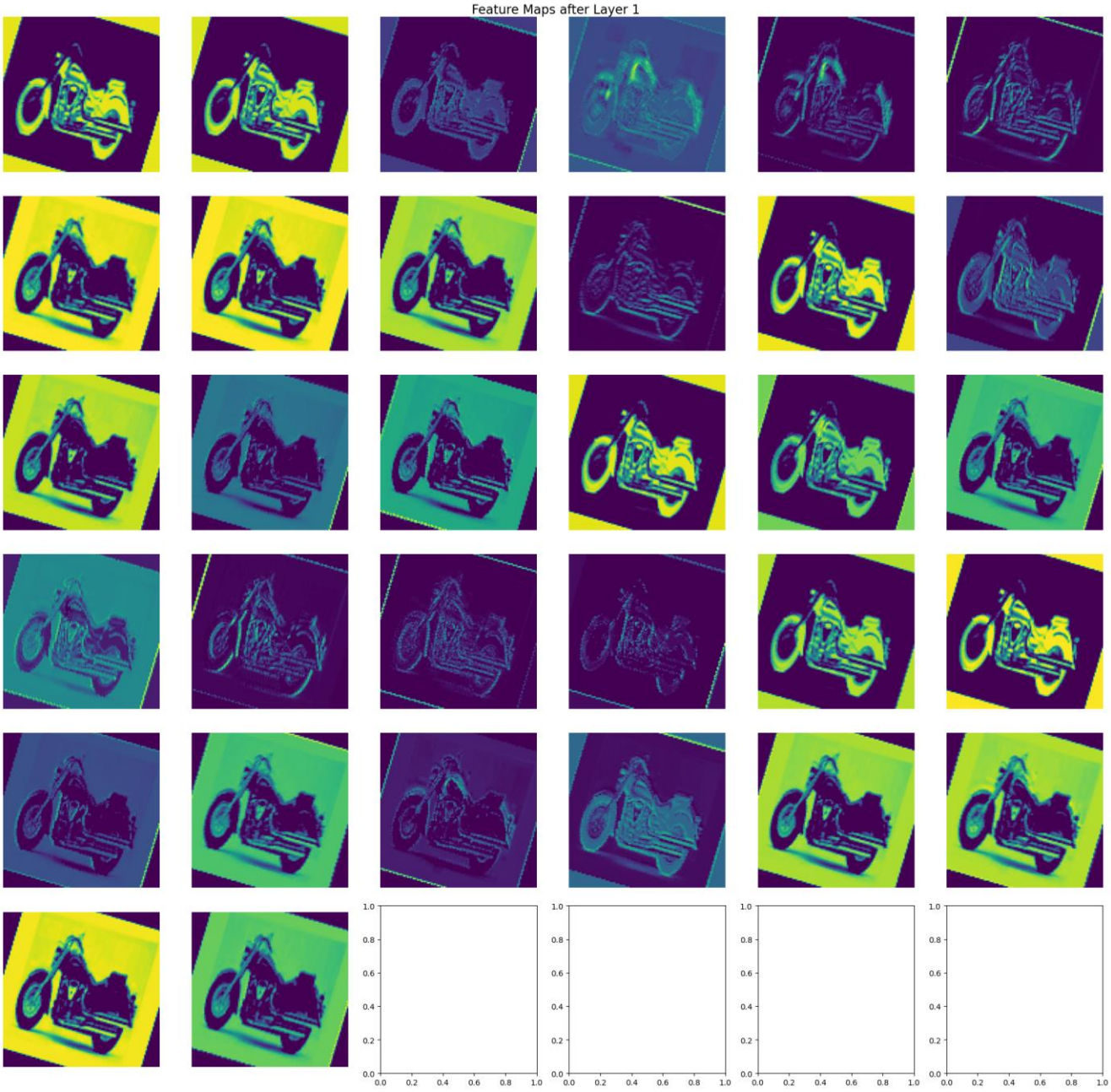

Top 5 Accuracy for Caltech-101_Deep
pooling
Max Pooling
Avg Pooling
Top 5 accuracy (%)
0
10
20
30
40
50
60
70
80
CrossEntropyLoss + Adam
CrossEntropyLoss + SGD
Optimizer + Criterion

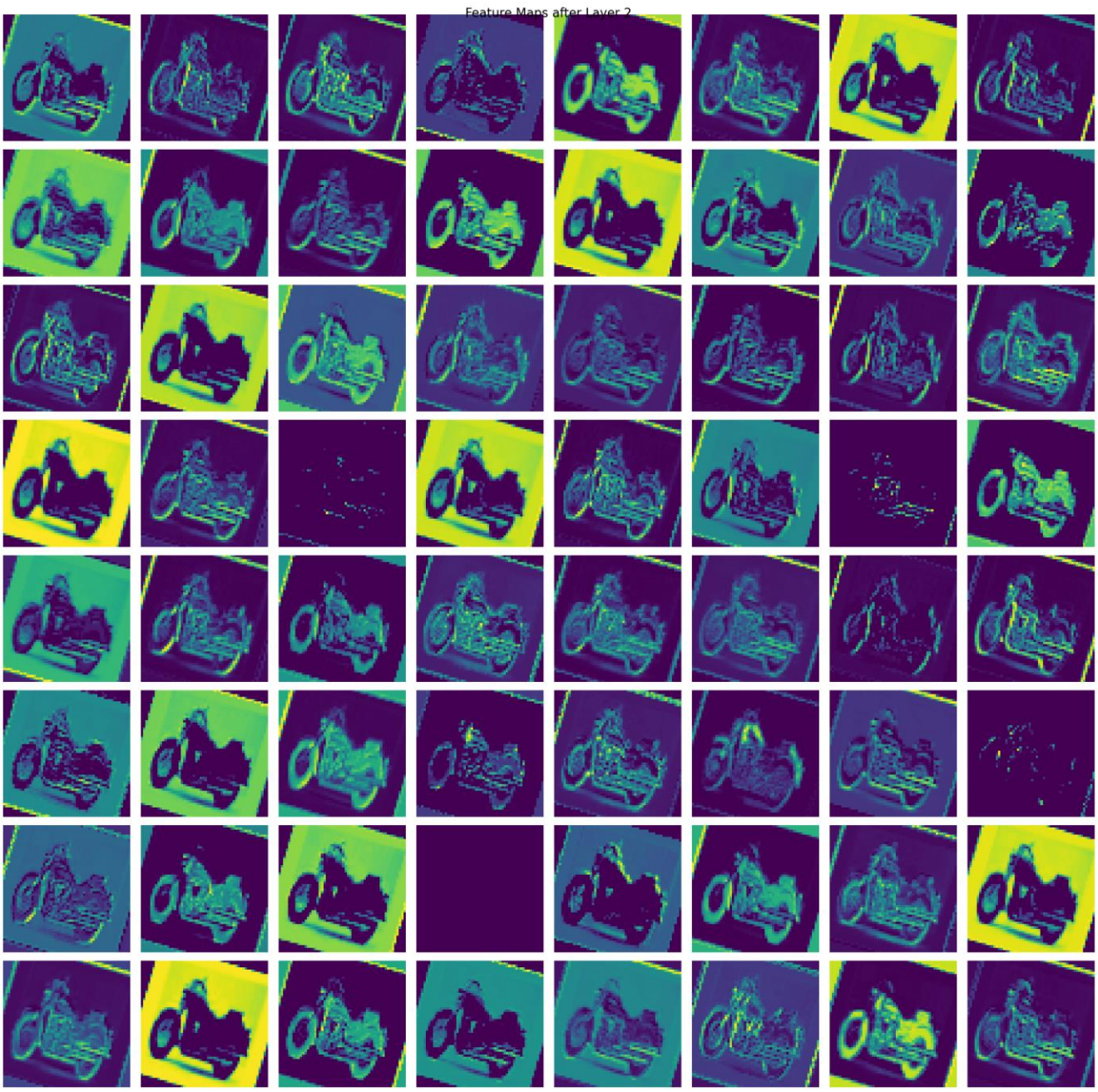
Feature Map Visualization

MNIST Feature maps

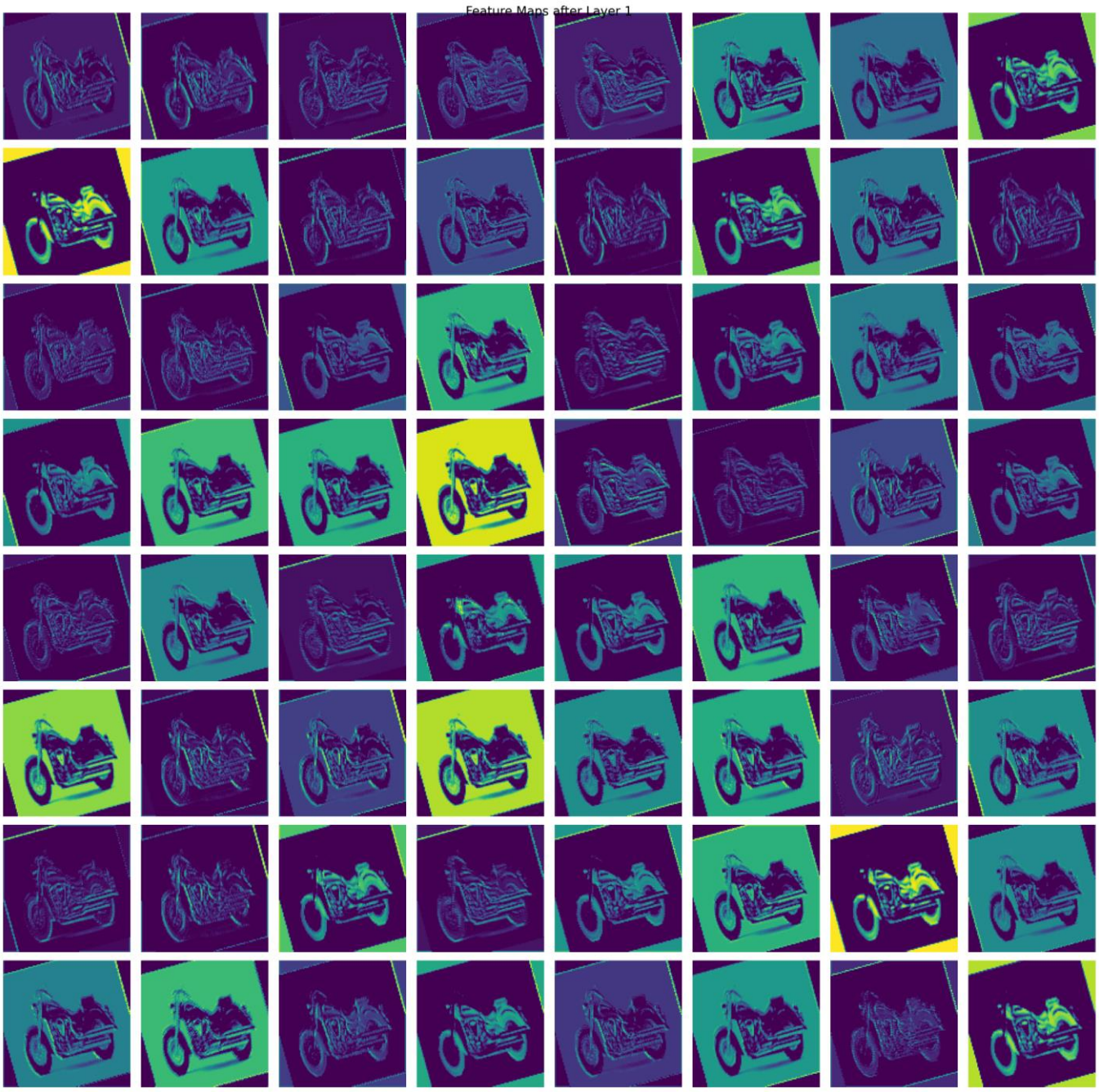

Feature Maps after Layer 2

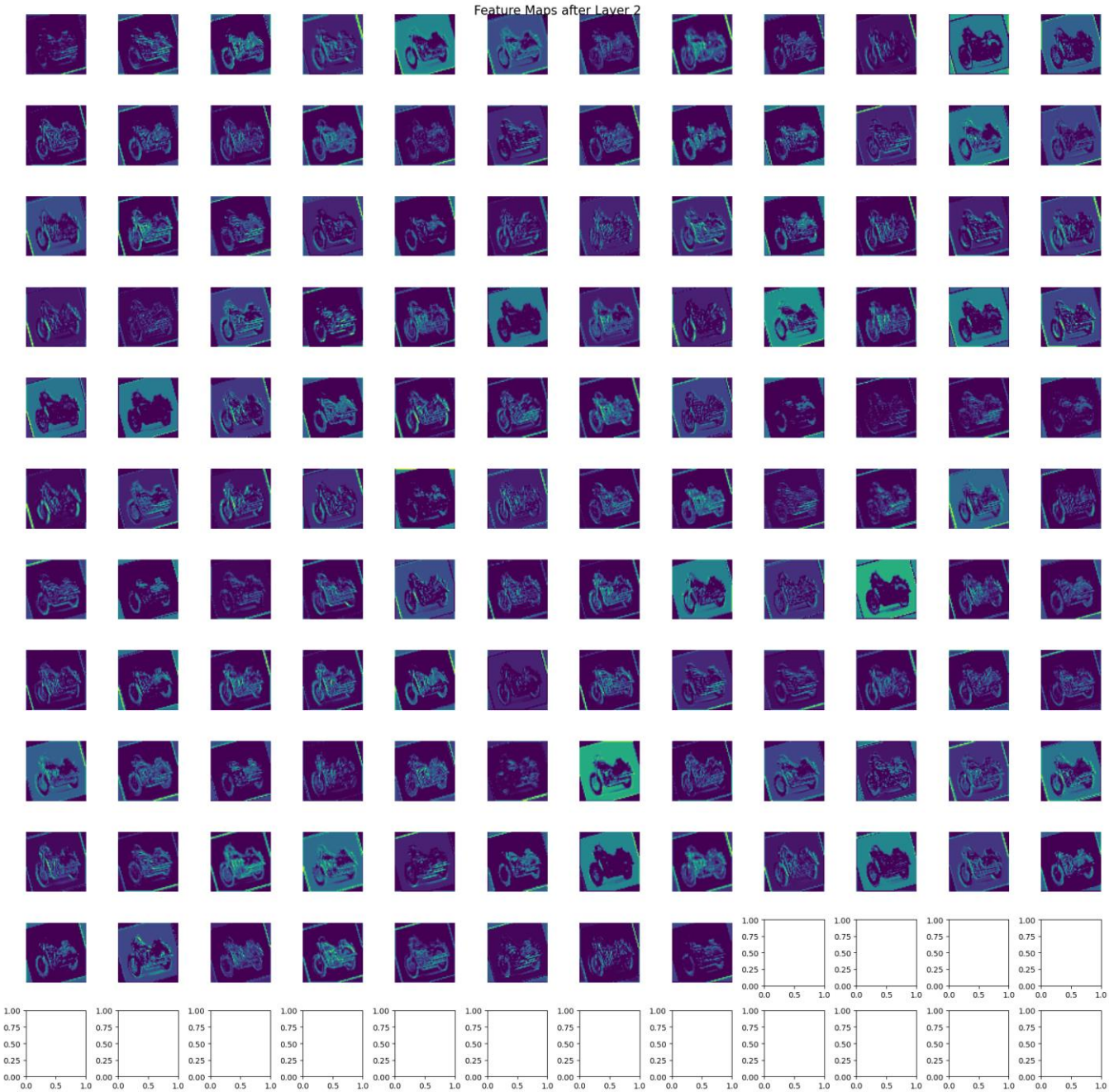
Caltech-101 Feature maps

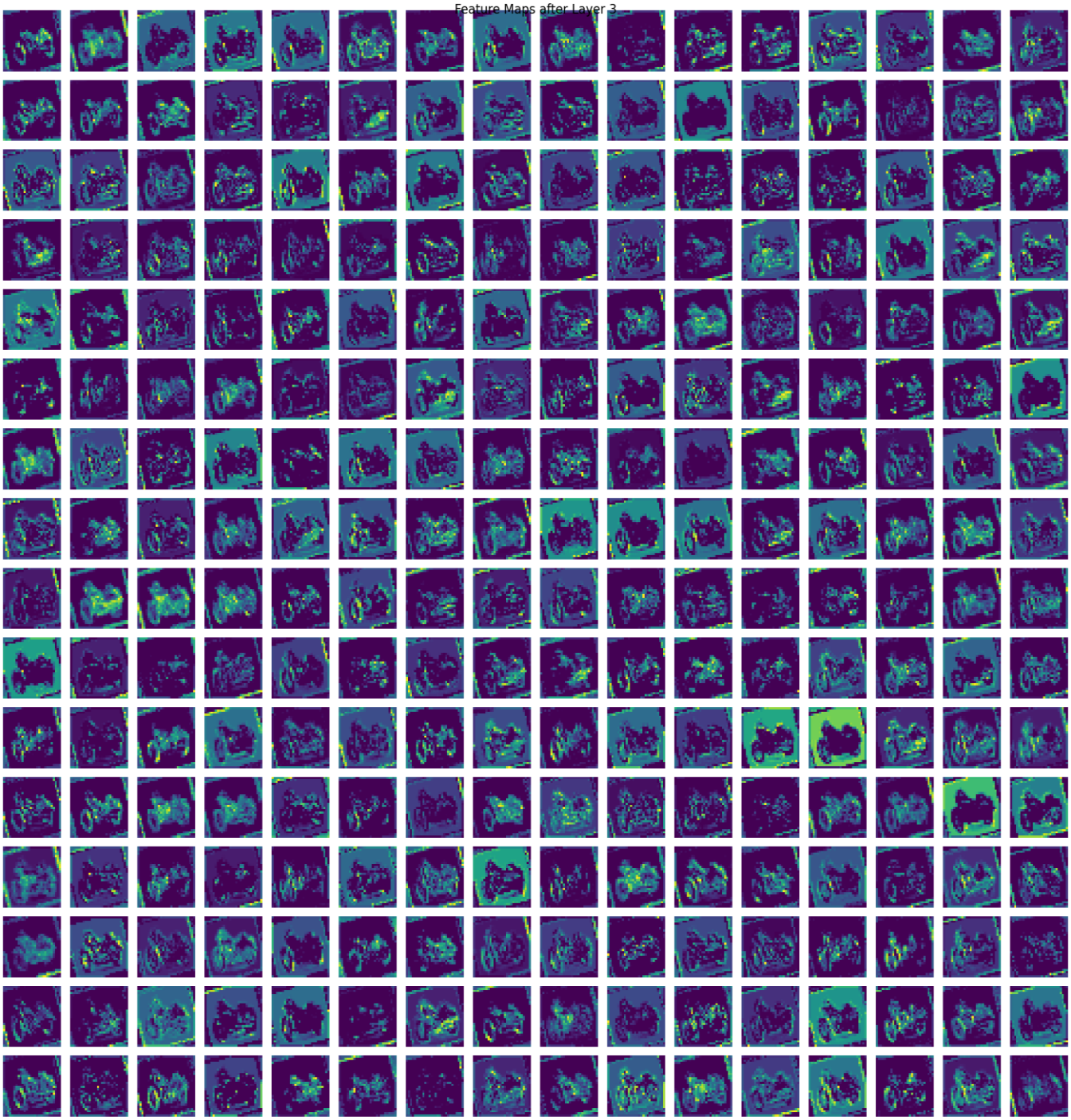


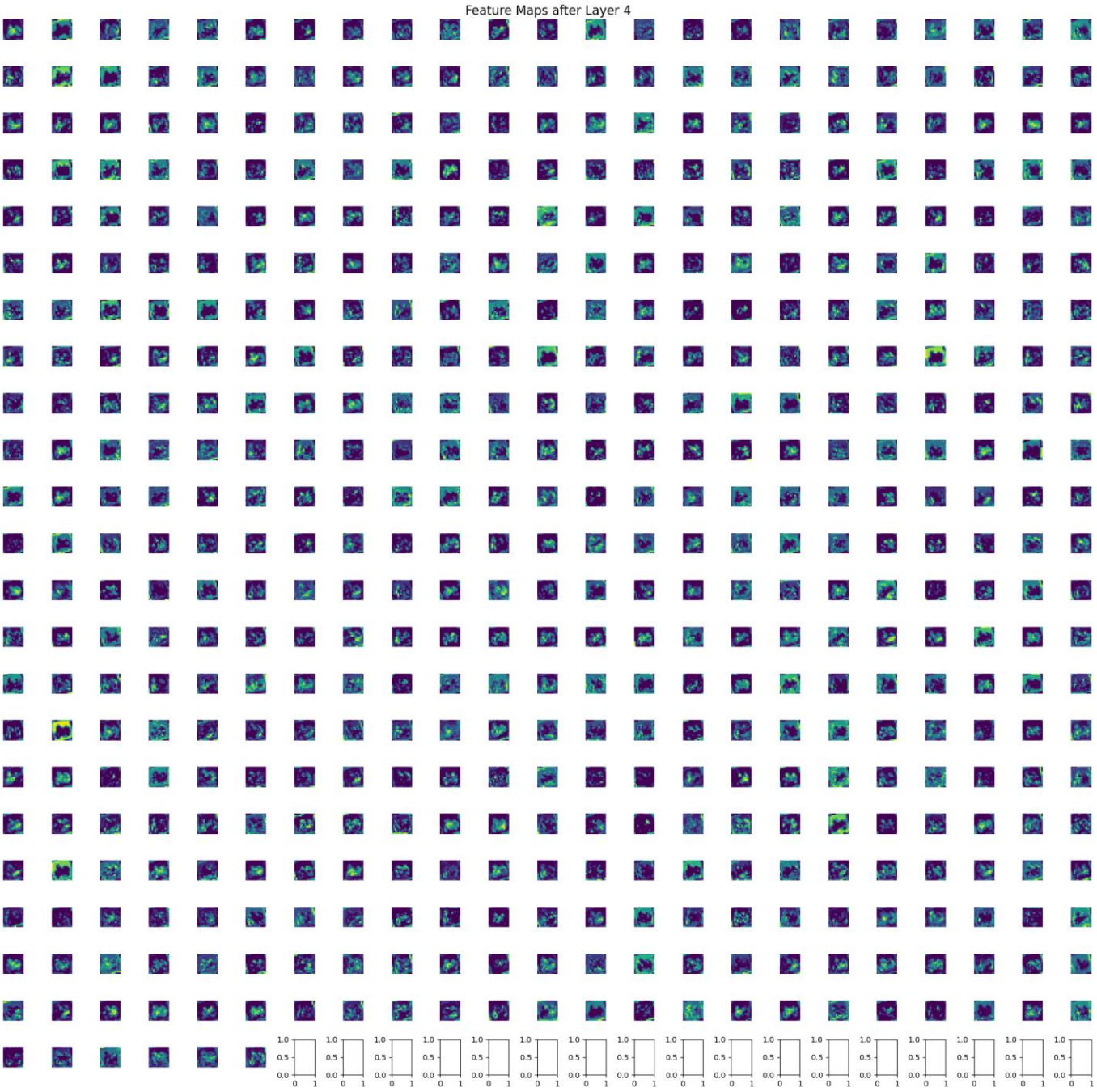

Feature Maps after Layer 2

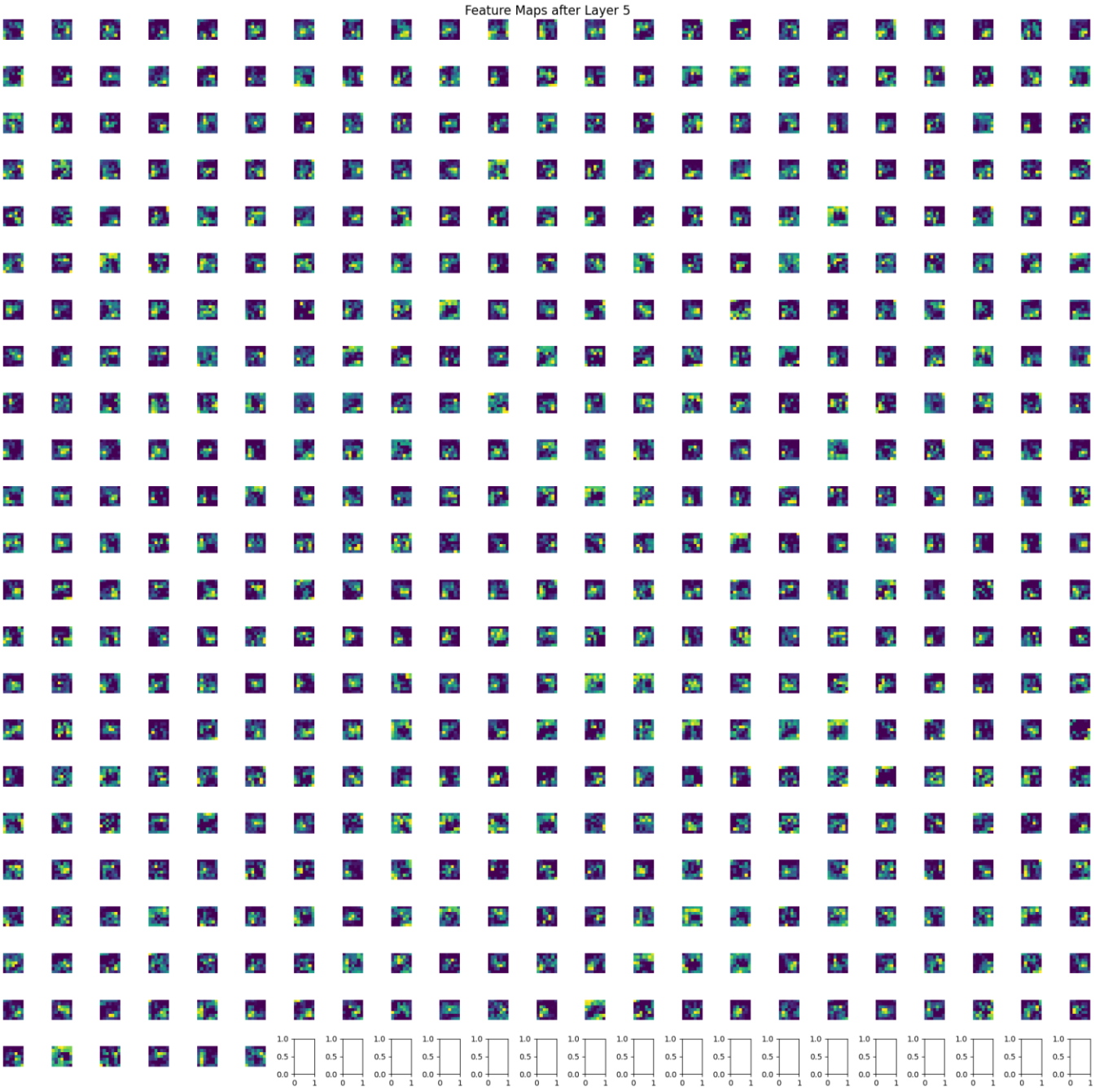
Caltech-101 Deep model Feature maps




Feature Maps after Layer 2


Feature Maps after Layer 3


Feature Maps after Layer 4


Feature Maps after Layer 5

MNIST Model Feature Maps Observations:

- After Layer 1:

The feature maps show clear detection of the digit's edges and contours. Each filter emphasizes different parts of the '9', with some maps highlighting the inner curves while others focus on the outer edges. Some maps detect horizontal or vertical line segments, indicating that the first layer has learned basic geometric shapes and patterns. The intensity is relatively uniform across maps, signifying that most feature maps are contributing to detecting the essential patterns of the digit.

- After Layer 2:

The feature maps become sparser and much less pronounced, with only fragments of the digit '9' remaining in many maps. Some feature maps appear almost blank, while others focus on very small parts of the digit (such as its top curve or the lower loop). The diminishing detail indicates that the second layer is highly selective, focusing on more abstract patterns, possibly filtering out noise or unimportant features.

MNIST Model Feature Maps Analysis:

- Layer 1:

The first layer in this CNN acts as a basic edge detector. Different filters capture varying orientations and shapes, enabling the model to generalise better across the dataset. In MNIST, composed of simple digit images, these edge and shape detections suffice for distinguishing one digit from another. Layer 1 is crucial in creating the building blocks for further abstraction.

This works because MNIST is a relatively simple dataset, so the initial feature extraction layer can focus primarily on edges and basic contours. This is enough to establish a strong foundation for later decision-making in the network.

- Layer 2:

This layer refines the previous information, discarding unnecessary parts of the image. It shows increased selectivity, focusing on specific regions of the digit. The sparsity suggests that the network is learning to emphasise the most critical aspects of the digit. For example, while the full digit contour is captured in Layer 1, only specific parts of the '9' are considered significant in Layer 2.

This increased selectivity allows the network to differentiate between subtle variations in digits, reducing the risk of overfitting by disregarding irrelevant parts of the input. This refinement is essential for ensuring that the network can handle test data it hasn’t seen before.

Caltech-101 Model Observations:

- After Layer 1:

The first layer of the Caltech-101 model captures more complex patterns than MNIST. The feature maps highlight major contours of the motorcycle (such as the wheels, body frame, and handlebars). Each filter is specialised for detecting different aspects of the object, but most feature maps still preserve the broader image. There is more variety in the colours of activation compared to the MNIST model, indicating that the network is dealing with a more diverse and detailed input.

- After Layer 2:

The feature maps after Layer 2 show increased abstraction. Specific parts of the motorcycle (like the wheels or the frame) are being highlighted in different filters. The network begins to focus on smaller, more detailed sections, such as the curves of the wheels or the motorcycle’s frame structure.

Caltech-101 Model Analysis:

- Layer 1:

Like the MNIST model, this first layer is dedicated to extracting basic edges and contours. However, because the Caltech-101 dataset is more complex, the feature maps focus on larger, more intricate parts of the image, such as the entire motorcycle frame. The filters handle both edges and texture, with some maps capturing more abstract contours like shading and depth.

Why this step is crucial: In a complex dataset, this broader range of feature detection allows the model to handle objects of varying shapes and sizes. The diversity of filters ensures that the model doesn't just memorize specific shapes but generalizes across different object categories.

- Layer 2:

The second layer refines the initial feature maps, isolating key details of the motorcycle while filtering out irrelevant background information. This layer abstracts the object into more specific components (e.g., focusing on the wheel's curvature in one filter, or the motorcycle's frame in another). Unlike MNIST, this layer is more concerned with textures and finer details, as seen in the maps that emphasize internal parts of the object.

For the more complex Caltech-101 dataset, a finer level of detail is needed to differentiate between objects. By focusing on smaller aspects of the object, the network can capture the most discriminative features required for accurate classification. This refinement ensures that the network can handle multiple objects and complex scenes.

Caltech Enhanced Model Observations:

- After Layer 1:

The feature maps resemble those from the basic Caltech-101 model but are more detailed, given that more filters are employed. This allows for better capture of various aspects of the image. The maps show a mix of both broad outlines and fine-grain texture detection (such as shadows or reflections on the motorcycle). The filters are much more diverse, capturing everything from the wheels to the background elements in the image.

- After Layer 2:

The feature maps become more refined, isolating the most critical details of the object. More filters remain active compared to the original Caltech-101 model, indicating that the deeper architecture extracts richer features. Some maps focus on very small portions of the image, such as a small section of the wheel, while others capture the object's overall shape.

- After Layer 3:

At this point, the maps are highly abstract. The model has captured intricate details and now focuses on more abstract patterns like texture, depth, and shading. The maps look more complex, with finer granularity and attention to specific parts, such as the intricate details of the frame or reflections on the motorcycle.

- After Layers 4 & 5:

These layers focus on very sparse and abstract activations. Only small regions of the image remain highlighted, and most feature maps look much less detailed than the earlier layers. The deeper layers focus on very high-level abstractions critical for final classification. The maps seem almost uninterpretable regarding

raw pixel information, as they are far removed from the input image.

Caltech Enhanced Model Analysis:

- Layer 1:

The first layer is similar in purpose to the original Caltech-101 model but offers more complexity due to the increased number of filters. It captures more detailed aspects of the image, leading to a richer set of patterns to help later layers generalise better.

The additional filters provide more coverage of different aspects of the image, leading to a broader base of extracted features. This ensures that the deeper layers have a more complex set of inputs to build upon, resulting in better classification performance.

- Layer 2:

The second layer continues to refine the image, isolating more complex parts of the motorcycle, such as smaller textures or fine-grain details like reflections. This is more pronounced than in the original Caltech-101 model, indicating that the enhanced architecture performs a more thorough image analysis.

The additional depth allows for more nuanced feature extraction. By refining the images in this way, the model can generalize better and capture more discriminative features, making it easier to handle a wider variety of object categories.

- Layer 3:

At this point, the network is learning highly abstract features, as evidenced by the complex patterns seen in the feature maps. These are far removed from the pixel data and represent higher-level semantic information.

In deep learning, it is crucial that deeper layers form abstract representations. This abstraction allows the network to be invariant to slight changes in the input image, like position,

rotation, or illumination, which is particularly important for complex datasets like Caltech-101.

- Layers 4 & 5:

The last two layers show very sparse activations, indicating that the network focuses on high-level concepts. The maps in these layers are highly abstract and no longer tied to specific parts of the object but instead represent patterns that help in final classification.

These layers capture the most discriminative features while discarding unnecessary details. The fact that only small parts of the image remain active shows that the network has distilled the image into its most critical components, which is essential for accurate classification.

Comparison of Models:

1. MNIST vs. Caltech Models:

- The MNIST model deals with simpler patterns, evident from the more uniform and less abstract feature maps in both layers. The network does not need as much depth or complexity to achieve high accuracy.

- In contrast, the Caltech-101 and Enhanced models deal with more complex data and require deeper layers to extract and refine patterns. Their feature maps are more diverse, with different layers focusing on increasingly abstract parts of the image.

2. Caltech-101 vs. Caltech Enhanced:

- The original Caltech-101 model captures basic contours and textures but starts losing details by the second layer. On the other hand, the deeper Enhanced model can retain more details across layers, refining the image into higher-level abstractions.

- The Enhanced model shows more complex feature maps with a greater diversity of patterns in later layers, allowing it to handle more difficult cases better. It learns richer, more abstract representations, especially in the last layers, contributing to improved classification accuracy.

Final Thoughts and Future Work:

1. Model Design Process:

I initially designed simple CNN architectures for MNIST and Caltech-101 to establish a baseline performance and understand the core CNN mechanics across different datasets. These models enabled me to evaluate the effects of pooling strategies and optimizers on simpler tasks (MNIST) versus more complex tasks (Caltech-101).

2. Training Observations:

- MNIST Dataset: The model quickly converged, with near-perfect Top-1 accuracy. Adam optimizer demonstrated rapid convergence across both Max Pooling and Average Pooling methods, showing minimal variance between them for this simpler dataset.

- Caltech-101 Dataset: Initial models faced difficulties due to the higher complexity of the dataset. While Adam led to faster convergence, SGD with momentum and cosine annealing led to more stable performance in deeper training regimes, particularly when paired with Max Pooling.

3. Incremental Improvements:

- The enhanced model was derived from iterative improvements based on the performance of earlier models. Observing poor generalization in the initial Caltech-101 model, I designed a deeper network, incorporating dropout and batch normalization.

- Data Augmentation and Scaling: After analyzing performance, I incorporated various data augmentation techniques such as random rotations, horizontal flips, and color jittering. These were designed to increase the dataset diversity and prevent overfitting.

- Model Scaling: Scaling the model by adding more layers helped to improve feature extraction depth, handling more intricate features such as textures and object parts, which were critical in improving classification accuracy on Caltech-101

4. Feature Map Insights:

- MNIST: Even simple CNN architectures captured fundamental patterns like edges and curves. Feature maps from early layers demonstrated how CNNs extract low-level visual patterns critical for classification.

- Caltech-101: In deeper layers, the enhanced model successfully learned increasingly complex and abstract features, including object shapes and textures. This reinforced the importance of depth and regularization for complex image datasets. Max Pooling enabled effective feature preservation across deeper layers, further aiding classification accuracy.

5. Validation of CNN Theory:

- The study confirms the CNN principle that deeper networks learn more abstract and sophisticated features, vital for classifying intricate images. Dropout and batch normalization were key in preventing overfitting in these deeper models, confirming theoretical assumptions on regularization techniques.

- By leveraging feature maps, data augmentation, and scaling techniques, I demonstrated how CNN architectures can be iteratively improved, aligning with CNN theory's emphasis on hierarchical feature learning and pattern abstraction.

6. Future Work:

- Fine-tuning Hyperparameters: Exploring additional optimizers (e.g., RMSprop) or advanced learning rate schedules might further stabilize and enhance the model performance on Caltech-101.

- Advanced Data Augmentation: Techniques such as CutMix or MixUp could be introduced to artificially increase dataset diversity and further reduce overfitting.

- More Complex Regularization: Incorporating techniques like weight decay, early stopping, or even adversarial training could improve robustness and generalization further.

Conclusion:

This work shows how CNN theory, when applied systematically, enables significant performance improvements in image classification tasks. Starting with simple models, I iteratively applied concepts like data augmentation, deeper architectures, dropout, and batch normalization to improve model performance on complex datasets. By refining hyperparameters and exploring model regularization.