

CSCI 611 - Assignment 2

Olivia Johnson

20 February 2026

1 Training and Visualizing a CNN on CIFAR

1.1 Model Architecture and Training Setup

1.1.1 Brief Description of Architecture

The CNN takes as input a $32 \times 32 \times 3$ RGB image from the CIFAR-10 dataset and processes it through three convolutional blocks followed by a fully connected classification head.

Convolutional Feature Extractor

The network contains three convolutional layers, each using a 3×3 kernel with stride 1 and padding 1. Padding preserves the spatial resolution after convolution.

- **Conv1:** 3 input channels \rightarrow 16 output feature maps
Output size: $32 \times 32 \times 16$
- **Conv2:** 16 input channels \rightarrow 32 output feature maps
Output size: $16 \times 16 \times 32$ (after pooling)
- **Conv3:** 32 input channels \rightarrow 64 output feature maps
Output size: $8 \times 8 \times 64$ (before final pooling)

Each convolutional layer is followed by:

- ReLU activation to introduce nonlinearity
- Max pooling (2×2 , stride 2) to reduce spatial dimensions by half

The spatial resolution changes as follows:

$$32 \times 32 \rightarrow 16 \times 16 \rightarrow 8 \times 8 \rightarrow 4 \times 4$$

After the third pooling layer, the feature tensor size becomes:

$$64 \times 4 \times 4$$

These convolutional layers enable the network to learn hierarchical feature representations. Early layers detect low-level patterns such as edges and textures, while deeper layers capture more complex shapes and object structures.

Fully Connected Classification Head

The flattened feature vector ($64 \times 4 \times 4 = 1024$ features) is passed into two fully connected layers:

- **FC1:** $1024 \rightarrow 100$ neurons with ReLU activation
- **FC2:** $100 \rightarrow 10$ neurons (one output per CIFAR-10 class)

The final layer produces raw logits, which are used by the CrossEntropyLoss function during training.

1.1.2 Loss Function

I chose to use the PyTorch CrossEntropyLoss function. Cross Entropy Loss is a good choice when training a model for image classification because it effectively measures the difference between true and predicted probability distributions for classification. It penalizes confidently incorrect predictions, which helps guide the model toward accuracy.

The loss function is shown in the code snippet below:

```
criterion=nn.CrossEntropyLoss()
```

1.1.3 Optimizer

I chose to use the Adam (Adaptive Moment Estimation) optimizer over the SGD (Stochastic Gradient Descent) optimizer because I trained and tested with both and consistently saw better results with the Adam optimizer.

The optimizer is shown in the code snippet below:

```
optimizer = optim.Adam(model.parameters(), lr=0.001, weight_decay=1e-4)
```

1.1.4 Learning Rate

The learning rate with which I trained was 0.001, and I found that this was a value that allowed my training to run in a reasonable amount of time, while still achieving acceptable accuracy.

1.1.5 Batch Size

I kept the provided batch_size of 20.

1.1.6 Epochs

I trained with 20 epochs with a learning rate of 0.0001, then I observed on my plot that the curves meet around the 12th epoch. So, I switched to training with 12 epochs and observed a very similar accuracy to the model trained with 20 epochs. Then, I changed the learning rate to 0.001 instead of 0.0001 and ran my training again with 12 epochs and saw the curves intersecting around 5 epochs. I trained again with 5 epochs and saw a higher accuracy. The final number of epochs that I trained with was 5 epochs.

1.1.7 Regularization and Data Augmentation

I use a few different kinds of regularization in my code. First, in my optimizer, I added 'weight_decay = 1e-4'. This is L2 regularization, which penalizes large weights, adds $\gamma||w||^2$ to the loss, encourages smaller, smoother weights, and helps reduce overfitting.

I applied a Dropout function:

```
self.dropout = nn.Dropout(0.25)
```

This dropout function is applied twice in forward as seen here:

```
x = self.dropout(x)          # after flatten
x = F.relu(self.fc1(x))
x = self.dropout(x)          # after fc1
```

The dropout function sets 25% of activations to zero and helps to reduce overfitting.

The last type of regularization in this architecture is an implicit regularization through early stopping. We only save the model when the loss decreases. This keeps the best-performing model to test and helps prevent overfitting.

I did not add any data augmentation. The transforms function used does not augment the data, it only rescales the pixels.

1.2 Final Test Accuracy

With the CrossEntropyLoss function, ADAM optimizer, learning rate of 0.001, batch size of 20, and 5 epochs, the final accuracy was 74%, as seen in Figure 1 below.

```
[87] ✓ 3.5s
... Test Loss: 0.768313

Test Accuracy of airplane: 77% (778/1000)
Test Accuracy of automobile: 83% (836/1000)
Test Accuracy of bird: 59% (594/1000)
Test Accuracy of cat: 59% (597/1000)
Test Accuracy of deer: 71% (710/1000)
Test Accuracy of dog: 60% (607/1000)
Test Accuracy of frog: 77% (771/1000)
Test Accuracy of horse: 82% (822/1000)
Test Accuracy of ship: 85% (859/1000)
Test Accuracy of truck: 85% (852/1000)

Test Accuracy (Overall): 74% (7426/10000)
```

Figure 1: Test Accuracy - 5 Epochs

With the CrossEntropyLoss function, ADAM optimizer, learning rate of 0.001, batch size of 20, and 12 epochs, the final accuracy was 73%, as seen in Figure 2 below.

```
np.sum(class_correct), np.sum(class_total))
[82] ✓ 3.5s
... Test Loss: 0.758371

Test Accuracy of airplane: 76% (768/1000)
Test Accuracy of automobile: 83% (836/1000)
Test Accuracy of bird: 62% (623/1000)
Test Accuracy of cat: 59% (595/1000)
Test Accuracy of deer: 68% (685/1000)
Test Accuracy of dog: 57% (579/1000)
Test Accuracy of frog: 81% (810/1000)
Test Accuracy of horse: 77% (775/1000)
Test Accuracy of ship: 87% (874/1000)
Test Accuracy of truck: 84% (843/1000)

Test Accuracy (Overall): 73% (7388/10000)
```

Figure 2: Test Accuracy - 12 Epochs

1.3 Training and Validation Loss Curves

In the following figures, you can see that when my model is trained over 12 epochs, the training and validation loss intersect somewhere between 5 and 6 epochs. I started with many more epochs and worked my way down to find the optimal number to train with.

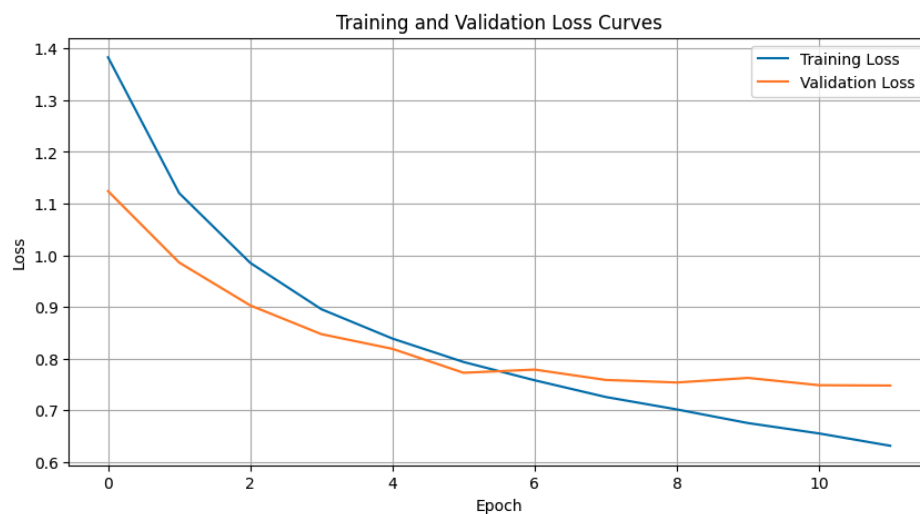


Figure 3: Training and Validation Loss - 12 Epochs

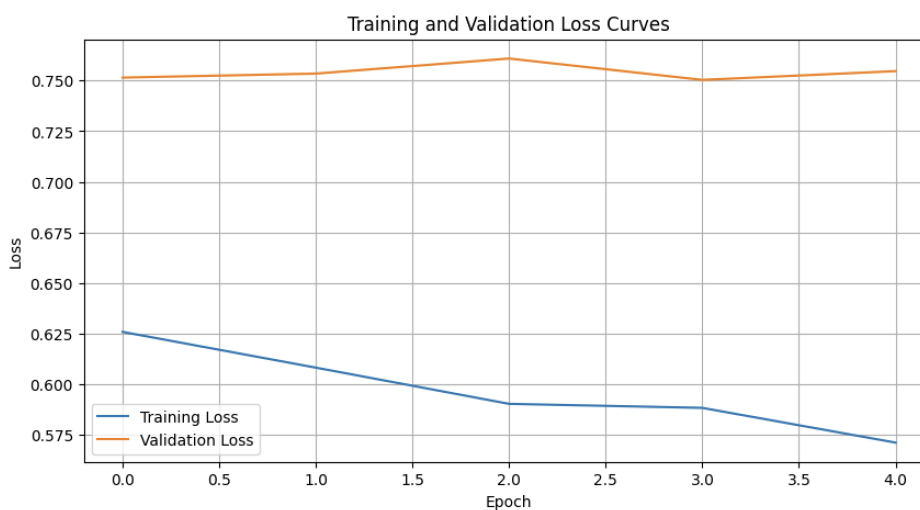


Figure 4: Training and Validation Loss - 5 Epochs

2 Feature Map Visualization

2.1 Feature Maps from the First Convolutional Layer

2.1.1 Image 1 - Ship

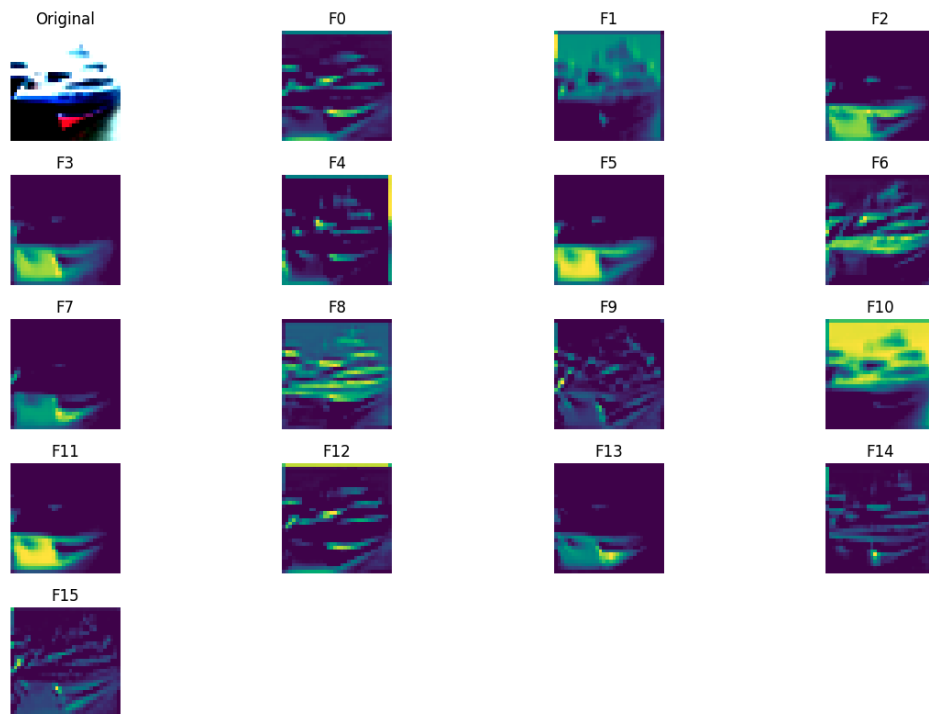


Figure 5: Ship Image - 16 Features

In the figure above, we can see the original ship image as well as the 16 features extracted by the first convolutional layer.

- F0: Response to the ship's mid-structure, likely a combined edge and texture detector.
- F1: Activates over the ship's upper region, could be a color-gradient detector.
- F2: Strong activation on bright hull area, likely a bright-region detector.
- F3: Mostly dark, not activated by this image.
- F4: Responds to ship body and some edges, likely a mid-frequency structure detector.
- F5: Strong response to bright lower-left region, likely a color contrast or bright blob detector.
- F6: Clear response along ship contours and diagonal structures, likely a diagonal edge detector.
- F7: Similar to F5 but slightly weaker, likely a localized brightness detector.
- F8: Strong activation over detailed ship structure, likely a high-frequency texture / horizontal edge detector.
- F9: Detects ship texture and patterns, likely a high-frequency texture detector.
- F10: Large smooth activation over bright sky/water region, likely a low-frequency intensity detector.
- F11: Responds to lower-left bright area, likely another intensity blob detector.
- F12: Responds to repeated horizontal patterns, likely a horizontal edge detector.
- F13: Localized bright-region detection, probably a small blob detector.
- F14: Strong response along horizontal ship lines, likely a horizontal edge detector.
- F15: Activates on fine details and edges, likely a texture + diagonal edge detector.

2.1.2 Image 2 - Horse

In the figure above, we can see the original horse image as well as the 16 features extracted by the first convolutional layer.

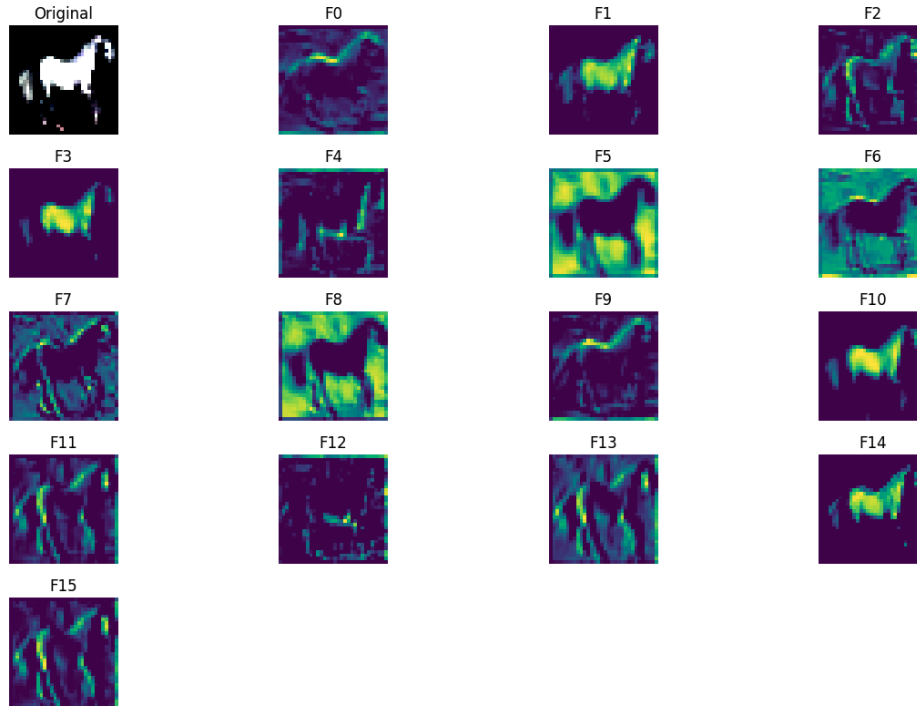


Figure 6: Horse Image - 16 Features

- F0: Responds to subtle background and mid-level texture, likely a low-contrast texture detector.
- F1: Strong activation tightly aligned with the horse silhouette, likely a shape detector.
- F2: Activates on vertical structures and background texture, likely a vertical edge detector.
- F3: Strong localized response on horse torso, likely a bright-object detector.
- F4: Highlights background vertical structures, likely a vertical edge + texture detector.
- F5: Strong full-body activation, likely a mid-frequency structure detector capturing overall shape.
- F6: Activates across entire silhouette and background, likely a broad contrast detector.
- F7: Responds to background clutter and contour transitions, likely a texture-sensitive filter.
- F8: Strong activation over horse body and legs, likely a global structure detector.
- F9: Activates on surrounding foliage/texture, likely a high-frequency texture detector.
- F10: Very localized activation centered on horse body, likely a small-scale bright-region detector.
- F11: Responds strongly to vertical background lines, likely a vertical edge detector.
- F12: Sparse and low activation, likely weakly aligned to this image.
- F13: Highlights vertical textures and contour fragments, likely a vertical + texture detector.
- F14: Very strong, clean activation of horse silhouette, likely a refined object-shape detector.
- F15: Activates on fine vertical edges and background structure, likely a high-frequency vertical edge detector.

2.1.3 Image 3 - Dog

In the figure above, we can see the original dog image as well as the 16 features extracted by the first convolutional layer.

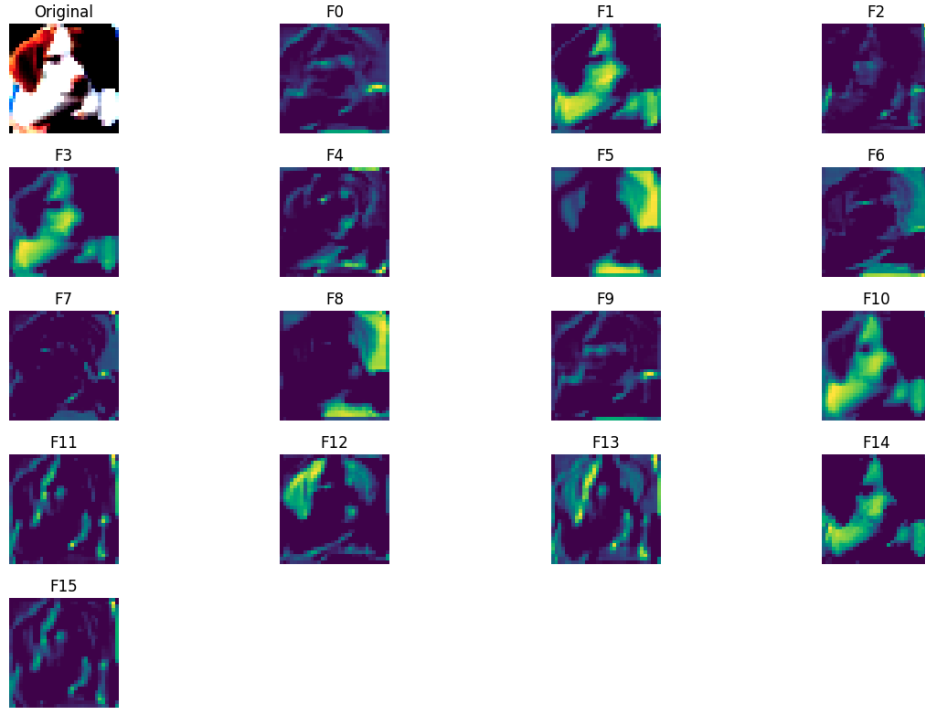


Figure 7: Dog Image - 16 Features

- F0: Responds to fine fur texture and subtle contour transitions, likely a high-frequency texture + weak edge detector.
- F1: Strong activation along the dog's torso and head contour, likely a mid-frequency structure detector.
- F2: Sparse activation on darker regions, likely a shadow-sensitive or low-intensity detector.
- F3: Strong response on bright chest and face, likely a bright-region / intensity blob detector.
- F4: Activates across fur details and internal gradients, likely a texture-sensitive filter.
- F5: Strong activation on high-contrast boundary (body vs background), likely a vertical edge or contrast detector.
- F6: Responds along curved body contour, likely a diagonal/curved edge detector.
- F7: Weak activation with scattered responses, likely a fine-grained noise/textured detector.
- F8: Strong activation on bright right side and boundary areas, likely a contrast-sensitive detector.
- F9: Highlights mid-level fur patterns, likely a mid-frequency texture detector.
- F10: Strong activation on torso and bright patches, likely a localized intensity detector.
- F11: Activates on vertical fur streaks, likely a vertical edge detector.
- F12: Responds to brighter upper regions and internal gradients, likely a structure-sensitive detector.
- F13: Strong activation on upper contour and bright regions, likely a diagonal edge + intensity detector.
- F14: Strong curved activation along body silhouette, likely a curved-edge detector.
- F15: Responds to fine fur edges and subtle structure, likely a high-frequency texture detector.

2.2 Maximally Activating Images for Selected Features

I defined the activation of a filter for an image as the maximum value in the feature map, which shows the strongest local response.

2.2.1 Filter 1

The top five images that maximally activate Filter 1 share a strong presence of bright pink or highly saturated color regions. In several of the images, the bright pink area occupies a large portion of the frame and creates strong contrast against darker surrounding regions. For example, the black lab wearing a bright pink bandana contains a highly saturated pink region in the lower-left quadrant, contrasted with the darker fur of the dog. Similarly, the bright pink car positioned in the center of a gray/black road produces a strong color contrast between the vivid pink object and the darker background. Another image includes a mostly dark or black-and-white subject with a bright pink region in the corner, again reinforcing the contrast pattern. The image of the cat lying on a bright pink blanket also follows this trend, where the saturated pink background dominates the scene.

One image deviates slightly from this pattern: the yellow puppy surrounded by bright pink. In this case, the pink region appears more as a background highlight rather than the primary object. However, the strong saturation and color contrast remain consistent with the other images.

Based on these observations, Filter 1 appears to function as a color-sensitive filter, likely responding to highly saturated regions, particularly pink or red hues, and strong color contrast boundaries. Rather than detecting a specific object class (e.g., dogs or cars), the filter seems to detect a general low-level visual feature related to bright color intensity and contrast. This suggests that the filter is capturing a general visual pattern rather than a class-specific semantic concept.



Figure 8: Top 5 Images for Filter 1

2.2.2 Filter 5

The top five images that maximally activate Filter 5 are all dominated by strong yellow coloration. In each case, a highly saturated yellow region occupies a significant portion of the image. For example, one image contains a yellow truck that fills most of the frame, while another shows a white cat positioned in the corner with the majority of the background consisting of bright yellow. A third image features an airplane viewed from above, where the full wingspan appears bright yellow, creating a large contiguous region of uniform color. The fourth image shows a vivid yellow frog centered in the image with a dark background, producing strong contrast. Similarly, the final image contains a bright yellow car against a predominantly black background.

Across these examples, two consistent patterns emerge: (1) large regions of saturated yellow color and (2) strong contrast between the yellow region and darker surroundings. This suggests that Filter 2 functions primarily as a color-selective filter, strongly responding to yellow hues and high color intensity rather than specific object categories. The fact that the filter activates for trucks, planes, frogs, and cars indicates that it is detecting a low-level visual feature (color and brightness) rather than a semantic object class. This behavior is characteristic of early- or mid-level convolutional filters, which often specialize in detecting specific color channels or color-contrast patterns.



Figure 9: Top Five Images for Filter 5

2.2.3 Filter 11

The top five images that maximally activate Filter 11 appear to share strong diagonal structures and corner-like features. Unlike the previous filters, which responded primarily to color, this filter seems to be responding to geometric structure.

In the first image, which contains a bird with a long beak, there is a clear diagonal line formed by the bird's body, as well as a sharp corner at the intersection of the beak and head. The second image, although difficult to identify precisely, contains several strong diagonal edges and angular transitions that likely contribute to high activation. The third image is more ambiguous and appears somewhat blob-like; however, subtle diagonal gradients or edge transitions may still be present and sufficient to activate the filter. The fourth image also contains distinct diagonal lines, even though the object itself is not immediately recognizable. Finally, the image of a horse's head shows a clear angular transition between the neck and nose, forming a corner-like structure, along with strong diagonal contours along the face.

Across these examples, the consistent pattern is the presence of oriented edges and angular intersections, particularly diagonal lines and corner formations. This suggests that Filter 3 functions as an edge-orientation or corner detector, responding to changes in intensity along specific directions rather than color information. Unlike Filters 1 and 2, which were strongly color-selective, this filter appears to capture a more structural, shape-based feature.



Figure 10: Top 5 Images for Filter 11

3 Brief Discussion and Reflection

In this assignment, I designed and trained a custom convolutional neural network on the CIFAR-10 dataset, achieving a final test accuracy of 74%. By tuning the learning rate and number of epochs, I observed that early stopping around 5 epochs provided strong performance while reducing overfitting.

Through feature map visualization, I observed that early convolutional layers learn interpretable low-level features such as edges, textures, color intensities, and contrast boundaries. The maximally activating image analysis further confirmed that individual filters specialize in detecting specific visual patterns, including color-selective responses (pink and yellow saturation) and geometric structures such as diagonal edges and corners.

Overall, this assignment strengthened my understanding of how CNNs progressively learn hierarchical representations, moving from simple visual features in early layers to more structured and meaningful patterns that support image classification.