Lab 2

Tristan Peat

October 2024

1 Introduction

In Lab 2, we examine the alignment between Artificial Neural Networks (ANN) and Spiking Neural Networks (SNN) trained on the CIFAR10 image classification task. The ANN architecture is a Convolutional Neural Network (CNN) with two Conv2d layers, followed by BatchNorm2d layers, ending with an MLP. The SNN follows a very similar architecture but with leaky integrate-and-fire neurons between each component. To explore the feature alignment we train a linear model to map between the CNN and SNN feature spaces (and vice versa) followed by an exploration of feature representation in AlexNet.

2 Symmetric Analysis of Linear Regression between CNN and SNN

Having trained a linear model mapping from SNN features to CNN features in class, we now explore the reverse mapping: from CNN features to SNN features. With the dataset and dataloader already created in class, the necessary code changes were 1) creating a new linear regression model to reinitialize the model parameters, 2) swapping which features the model makes predictions on 3) swapping the features the model uses for computing the loss from the objective criterion.

Training both models for 15 epochs with the same batch size, learning rate, we evalaute the mappings using linear regression loss, cosine similarity, and Euclidean distance. The results are summarized in Table 1.

Mapping Direction	**Test Loss**	**Euclidean Distance**	**Cosine Similarity**
SNN $\rightarrow$ CNN	0.2783	23.2826	0.7528
CNN $\rightarrow$ SNN	0.0221	6.6361	0.7068

Table 1: Comparison of mapping directions between SNN and CNN features.

The results indicate that the linear regression model mapping CNN features to SNN features achieved a lower average Euclidean distance compared to the

SSN to CNN mapping, suggesting that CNN features can be more accurately predicted from SNN features than vice versa. However, the average cosine similarity was higher for the SNN to CNN compared to CNN to SNN indicating a better alignment in terms of directional similarity in the SNN to CNN direction. These findings suggest that while linear mappings can partially align the feature spaces of SNNs and CNNs, the asymmetry in the mapping accuracy may be due to inherent differences in feature representations between the two network types.

We will next hypothesize reasons for the asymmetry.

1. CNNs typically produce continuous-valued activations, while SNNs generate spike-based representations that are inherently sparse and discrete. The spiking rates in SNNs may not capture the same level of feature detail as the activations in CNNs, making it harder to reconstruct CNN features from SNN features accurately.

2. The conversion of input stimuli into spikes in SNNs can lead to a loss of information due to the thresholding mechanism. This information loss may not be recoverable when mapping back to CNN features, resulting in higher test loss and Euclidean distance in the SNN to CNN mapping.

3. CNN features might have higher dimensionality and redundancy compared to SNN features (at least SNN have more sparse features because of the Leaky firing rate neuron); therefore a model might be able to exploit this sparsity when mapping from CNN to SNN. This might explain the better fit, judging from lower loss and Euclidean distance.

The higher cosine similarity in the SNN to CNN mapping indicates that, despite higher Euclidean distances, the directional trends of the features are better preserved. As the linear regression model never achieves a 0 loss and suffers from asymmetric differences, the mapping from SNN to CNN (and vise versa) lend itself to exploring using non-linear models for the transformation.

3 Kernel Visualization in AlexNet

The visualizations from the first Conv2D kernel are shown below in Figure 1. The cool part about the visualization of the earlier kernels is the channels are RGB therefore we can visualize color sensitivity of different kernels. We see some dominant colors such as red, purple, green, and blue (in addition to black and white of course). We also see evidence of edge detection, with many horizontal, vertical, or diagonal lines which are likely used by the model to determine rough object outlines. We also see several textured patterns. In some kernels we see sharp changes in color or contrast, which might serve to find changes in light or used to understand depth or separation in an image.

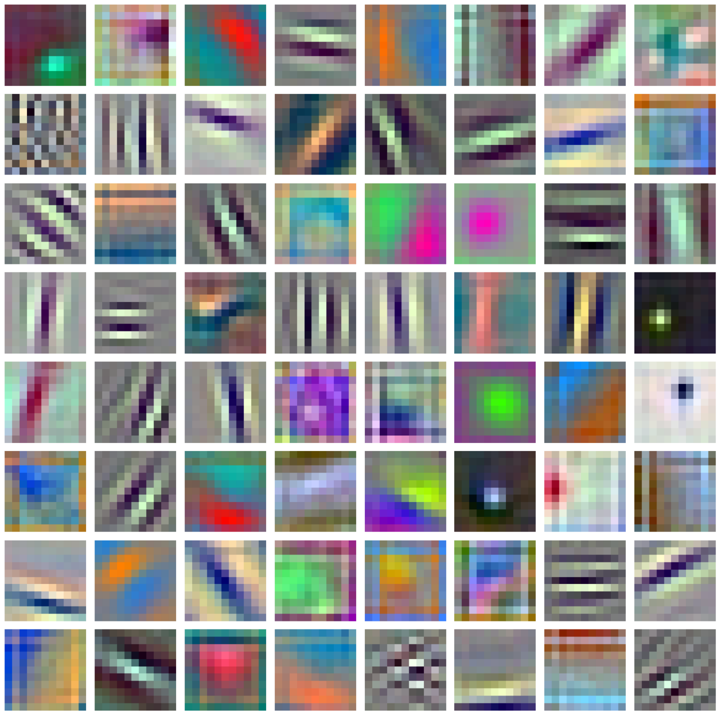

Figure 1: First kernel visualizations

The visualizations from the second conv kernel are shown below in Figure 2. While we are looking at a much lower resolution kernel, its still clear to see some unique patters of intensity and stripes (several bands of intensity in a row going some direction). We see kernels with the only intensity located right at the center point of the 5x5. We see kernels that spread the intensity from the center towards the edges with an even gradient. We see kernels that evenly split intensity right down the middle. Or kernels that gradient from left to right or top town. If we recall the how the Conv2d operation works, weighting the input values covered by a sliding kernel which is effectively feature extraction. So when we see unidirectional gradient-ed intensities, these might resemble the edge detection kernels. The intensity in the middle gradient-ed towards the edges might be the Gaussian blur kernel.

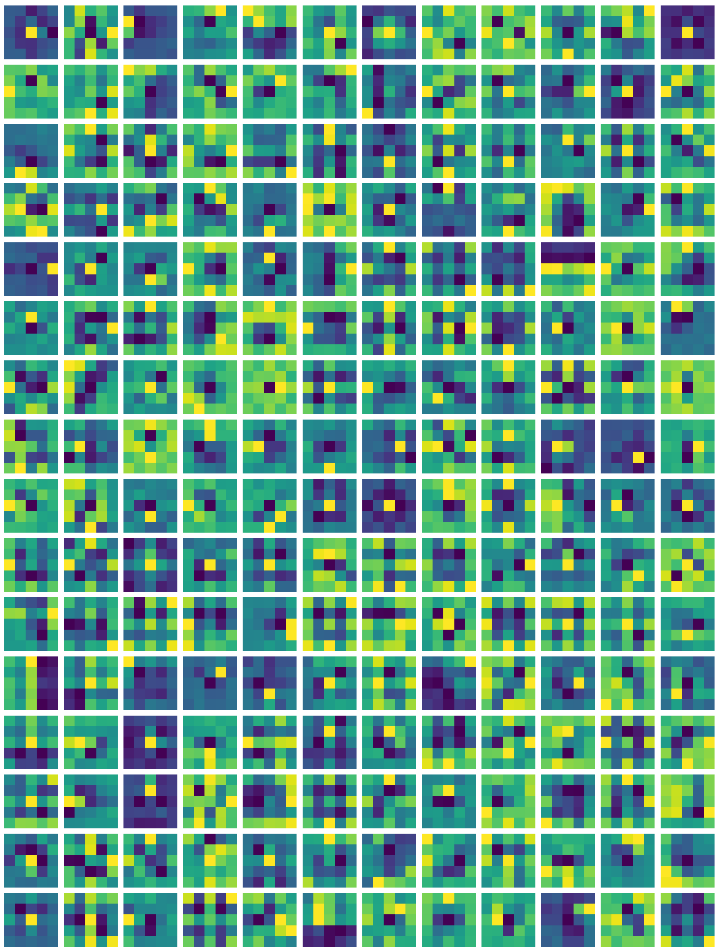

Figure 2: Second kernel visualizations

4 Visualization on the channel level

We will now compare the kernel visualization at a layer, channel, and neuron level; which is ordered by increasing granularity. We can expect to see larger shapes, edges, and colors at in the earlier layers.

For `layer` 0 in Figure 3, `cnn_filter_idx` 28 in Figure 4, and neuron x,y location $(10, 10)$ in Figure 5 we observe the following after 51 epochs. We see a vibrant jigsaw puzzle like image which can be inferred that the filters are responding to color and sharp edges. The color saturation is much higher in layer 0 then later layers because this convolution is acting directly on the RGB channels, which includes all color information. This information is semi-abstracted in later layers. The striped pattern is very distinct in channel 28 of layer 0 which might mean that the 28th channel is responsible for handling vertical edges or gradients. Neuron's in the first layer have a very small recpetive field (exactly 3x3 from the conv2d definition) and the visible vertical black stripe aligns with the black stripes observed in the channel 28 visualization at layer 0.

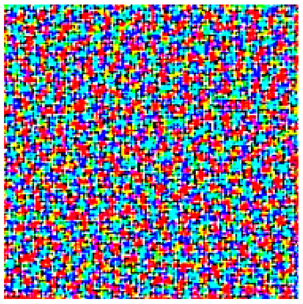
Figure 3: Layer 0

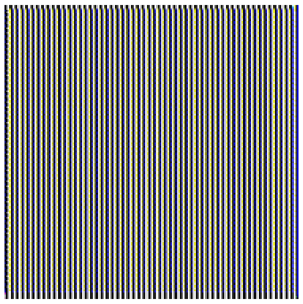
Figure 4: Channel 28

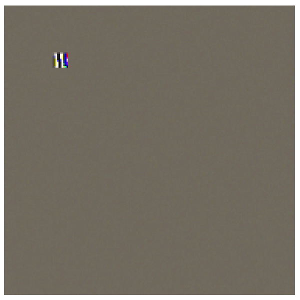
Figure 5: Neuron (10,10)

For `layer` 6 in Figure 6, `cnn_filter_idx` 28 in Figure 7, and neuron x,y location $(10, 10)$ in Figure 8 we observe the following after 51 epochs. Now we are one convolution layer deeper (technically the second convolution despite my naming convention of layer 6) and we can obvserve the basic features from layer 0 (conv1) being combined into more complex patterns. At layer 6, the channel 28 visualization now shows small colored dots which might suggest the filter responds to small distinct regions of the input. At layer 6, the neuron has an expanded broader view of the input, incorporating neighborhood information from previous layers. The neurons position now examines the bottom right area of the input image and its unique kleidoscopic shapes reflect the shapes seen in the channel visualization.

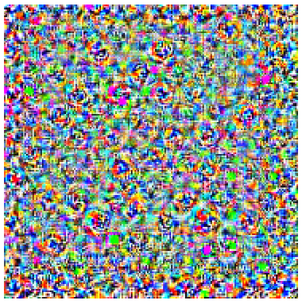
Figure 6: Layer 6

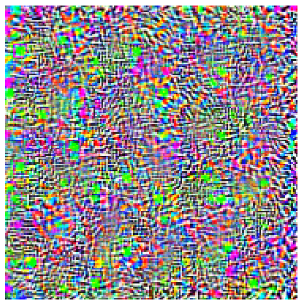
Figure 7: Channel 28

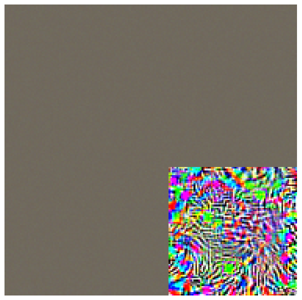
Figure 8: Neuron (10,10)

For `layer` 10 in Figure 9, `cnn_filter_idx` 28 in Figure 10, and neuron x,y location $(10, 10)$ in Figure 11 we observe the following after 51 epochs. At the layer level, larger swirls with vibration patterns can be seen which might represent complex structures of textures found in the ImageNet dataset. At layer 10, channel 28 we can see prism-like shapes forming in the visualization which might suggest the filter has now mutated to extract geometric shapes. The gray square seen in the neuron image shows that the neurons might not be sensitive to patterns anymore at this layer or combines information in a way that it simple averages it out.

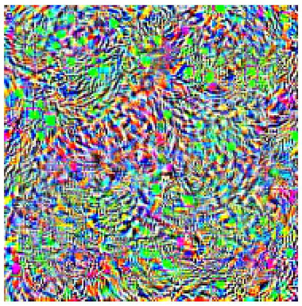
Figure 9: Layer 10

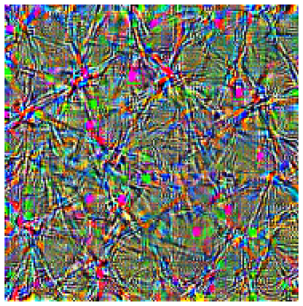
Figure 10: Channel 28

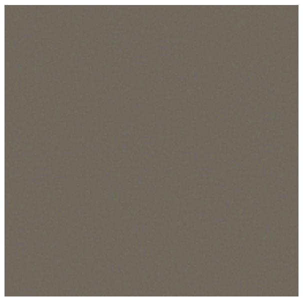
Figure 11: Neur(10,10)