COMP 4107 Assignment Five

Gabrielle Latreille (101073284)

Connor Stewart (101041125)

1. Modification of the dataset loaded:

   We load the dataset from TensorFlow Keras using the built-in aifar10.load_data() function in TensorFlow's Keras module. Keras formats the dataset as two tuples consisting of two variables each. The variables themselves represent the training and testing sets inputs and outputs, respectively. Finally, the application normalizes the data by dividing the training-set and testing-set input variables by 255 to produce a normalized version of the training and testing input parameters.

2. Changing the number of convolutional layers and sizes of max pooling layers. You must investigate 5 different model scenarios. Plot the network structures for each model using plot_model from tensorflow.keras.utils, and describe each network briefly:

   We use five different models in the project by adjusting parameters for the built-in Keras package in TensorFlow. Using Keras's models-function, we set the network parameters to produce the following models:

   Model One: One convolutional layer and one fully connected layer

   This model sets up a single convolutional layer (32 filters) followed by a max-pooling layer (with (2,2) strides) with a fully connected layer at the end, with all layers using relu activation. Output is generated using softmax activation and ten as the output space dimensionality.

   Summery: conv -> maxpool -> FC -> out

   Model Two: Two convolutional layers and one fully connected layer

   This model sets up a convolutional layer (32 filters) followed by a max-pooling layer (with two strides), with another convolutional layer (64 layers) followed by another max-pooling layer (with two strides) directly afterward. We add a third fully connected layer at the end, and all layers use relu

activation. Finally, the output is produced with a softmax activation and ten as output.

Summery: conv -> maxpool -> conv -> maxpool -> FC -> out

Model Three: Two convolutional layers and one fully connected layer

This model sets up a convolutional layer (32 filters) followed by another convolutional layer (64 filters), with a max-pooling layer (with two strides) right after. The model next gets a final fully connected layer. All layers use relu activation throughout the process to this point. The output is produced with softmax activation and ten as the output space afterward.

Summery: conv -> conv -> maxpool -> out

Model Four: Two convolution layers and two fully connected layers

A convolutional layer (32 filters) followed by a max-pooling layer (with two strides), with another convolutional layer (64 filters) followed by a further max-pooling layer (again with two strides) get added to the model. Afterward, two fully connected layers are added, with the first using a 128 as output and the second using 64 as output get added to the model. All the layers use relu activation up to this point. Output is produced using a softmax activation using ten as the output space.

Summery: conv -> maxpool -> conv -> maxpool -> FC -> FC -> out

Model Five: Four convolutional layers and three fully connected layers

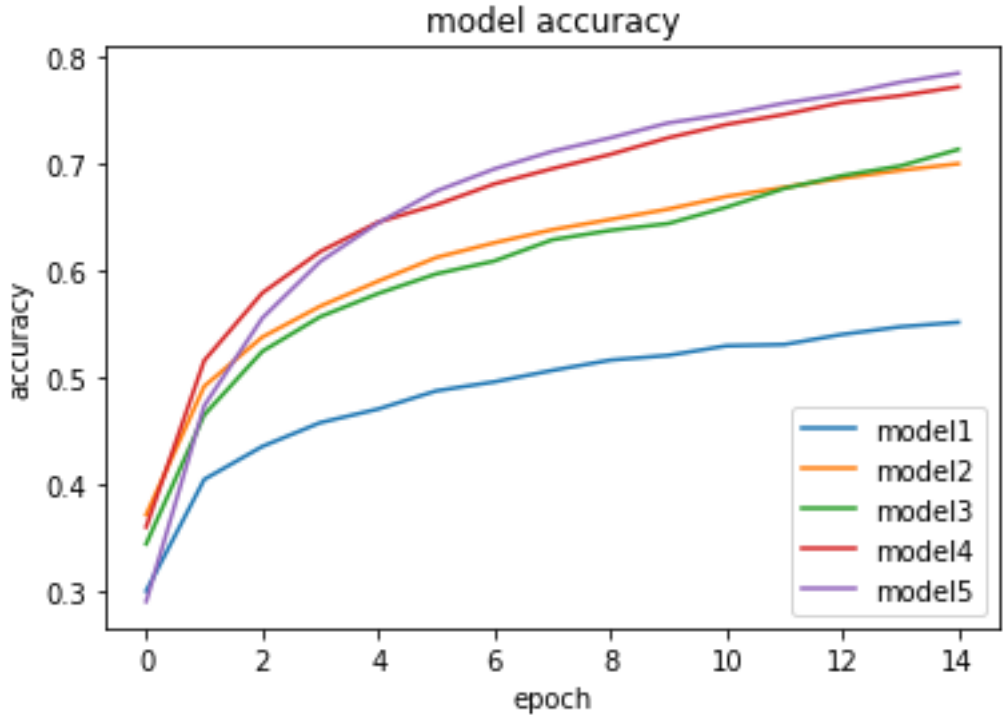
The final model uses a convolutional layer (32 filters) followed by another convolutional layer (64 filters) and then a max-pooling layer (with two strides). A third convolutional layer (64 filters) gets added, with a fourth convolutional layer (also 64 filters) getting initialized right after. A final max-pooling layer with strides of size two then gets added after the fourth convolutional layer indicated above. Finally, three fully connected layers get added - with 128, 64, and 32 as the output space dimensionality - respectively. All layers up to this point use relu activation, and a final output layer with ten as the dimensionality for the output space is produced, with the aid of a softmax activation function.

Summery: conv -> conv -> maxpool -> conv -> conv -> maxpool -> FC -> FC -> FC -> out

Note that all layers get flattened upon transitioning to the fully connected layers in all five models. All instances using a padding value of 'valid' when available, a kernel size of (3,3), and an input shape of (32, 32, 3) for the convolutional layers. Finally, a dropout of 0.25 is applied to the layers after each fully connected layer is generated.

3. Provide a chart of the accuracy of your network for 1-15 epochs for the scenarios investigated:

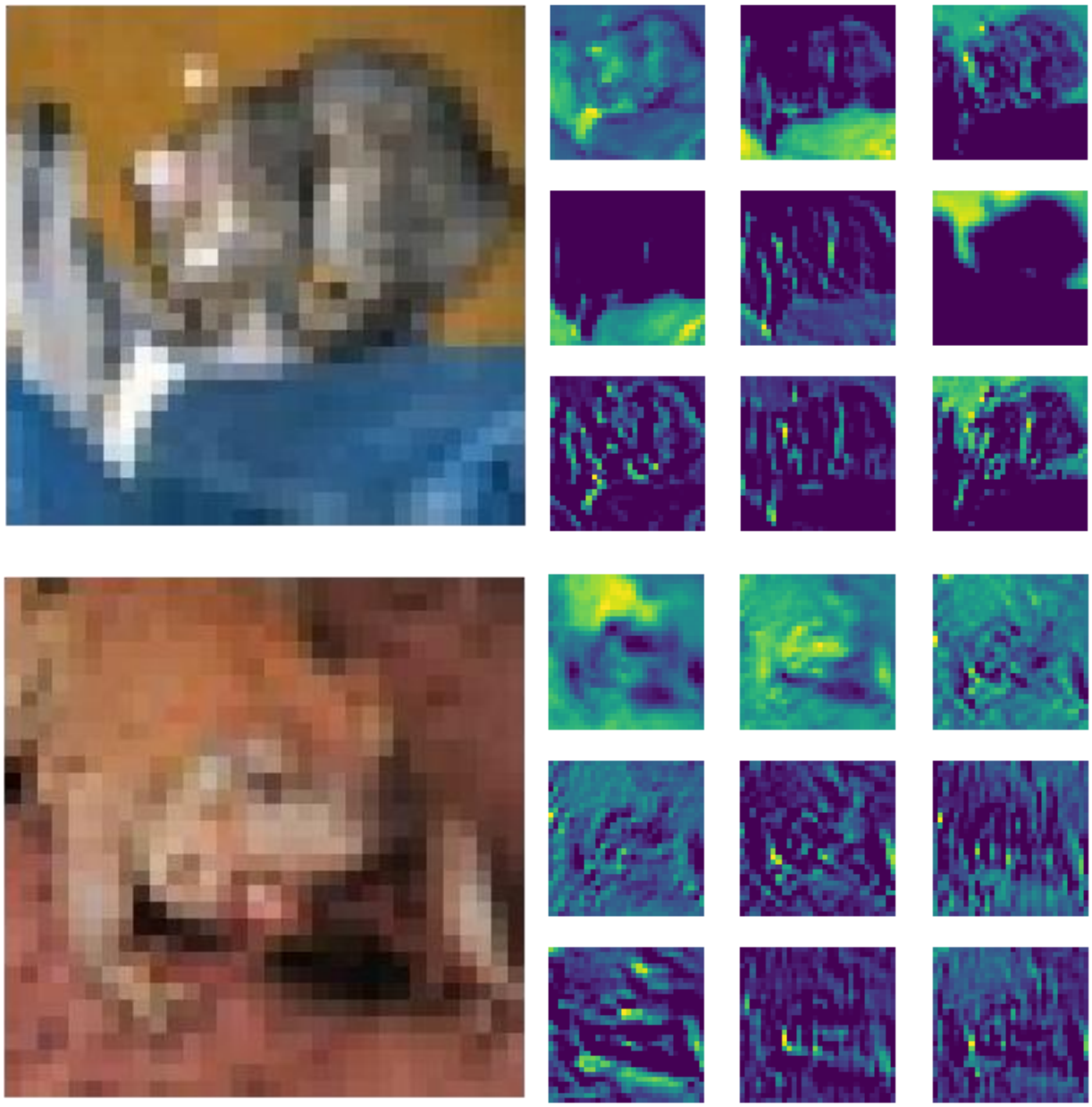
   The following are matplotlib graph representations of the accuracies for the five models shown above:



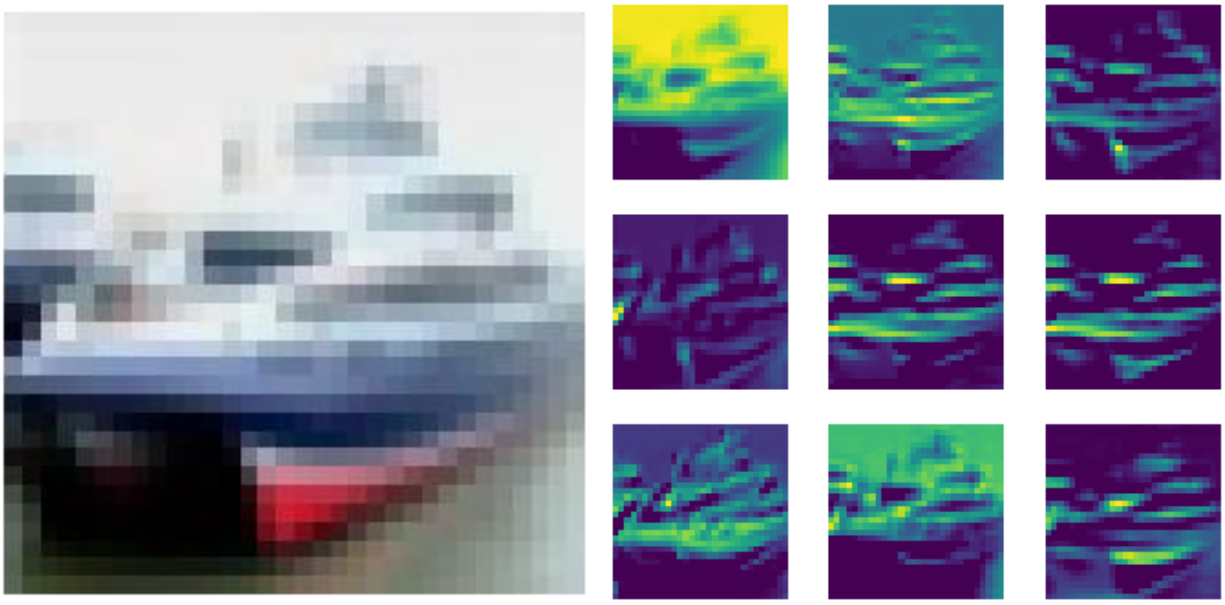
4. Provide the capability to show the top 9 patches (regions that best activate neurons), see slides 57-59 in CNN lecture (Lecture 16):

   The first model has two outputs, for image index zero and five from the aifar10 dataset. After that, we display the outputs for models 2, 3, 4, and 5 using images of index 1, 2, 3, and 4 from the aifar10 dataset. Displayed below, we see the top nine activated neurons (the nine small images on the right) next to their respective original picture from the dataset (the large image on the left):

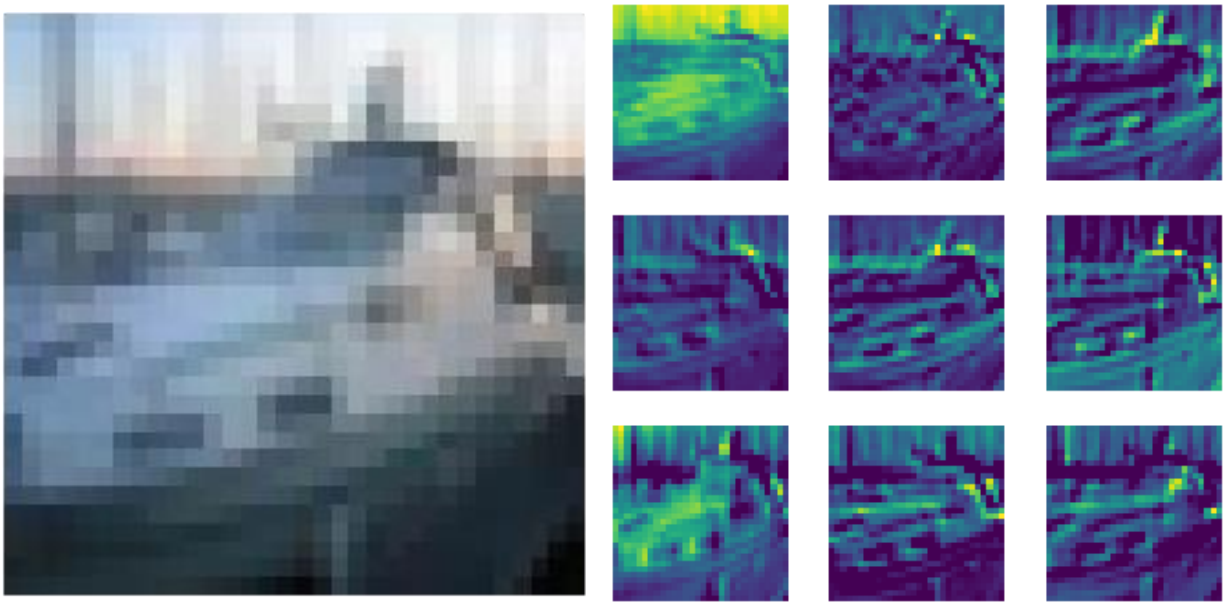
Model One:



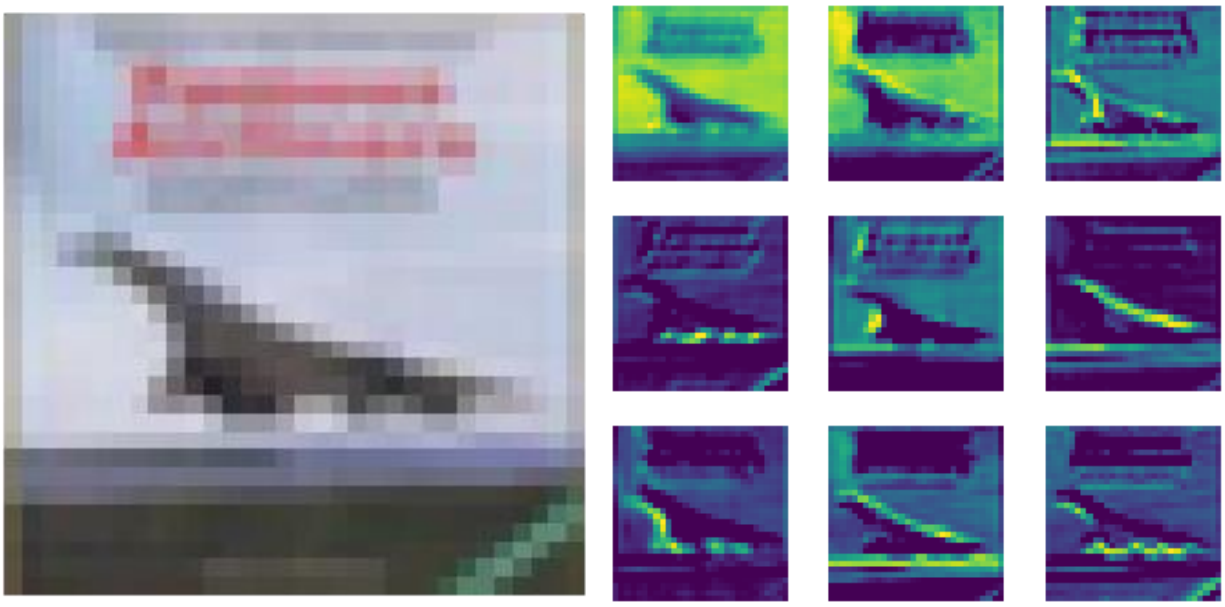
Model Two:



Model Three:



Model Four:



Model Five: