Image Classification using Convolutional Neural Networks

Paula Lozano Gonzalo

March 26, 2025

1 Introduction

This report shows the process of developing and evaluating convolutional neural network models for image classification tasks using the CIFAR-10 dataset. The goal is to train multiple neural network architectures, experiment with various hyperparam- eters, and get better than 0.75 accuracy.

2 Data Understanding

The CIFAR-10 dataset contains 60,000 small 32x32 pixel color images grouped into 10 distinct classes (e.g., airplanes, cars, birds). The data is split into 50,000 training images and 10,000 test images, with each class having exactly 6,000 images total. Classes are mutually exclusive—for example, "automobile" (cars) and "truck" (large trucks) are strictly separated.

Key observations:

- Images are low-resolution, which may limit fine details.

- The dataset is balanced, with equal representation across classes.

- Training data is divided into five batches, but each class retains 5,000 training images overall.

3 Data Preparation

Using the starter code file `open_data.py` we madde sure to separate the training from the testing and inside the training we also separated 10,000 images from the training data as validation data for the models to have a better idea of how the model predicts without showing it the testing data.

4 Model Architectures

(You can see the information that the model spits out on my `progress.md` file in the code)

The following models were trained:

4.1 Model A

This was only a test model, not even trained with the CIFAR-10 data but with the Fashion MNIST dataset. This was to make sure that the starter code worked from the begginin so that I could change it to the CIFAR-10 dataset after. The optimizer is `Adam(learning_rate=0.0001)`

```
+---------------------------------------+------------------------------+------------------+
| Layer (type)                          | Output Shape                 |          Param # |
+=======================================+==============================+==================+
| conv2d (Conv2D)                       | (None, 28, 28, 64)           |            3,200 |
+---------------------------------------+------------------------------+------------------+
| max_pooling2d (MaxPooling2D)          | (None, 14, 14, 64)           |                0 |
+---------------------------------------+------------------------------+------------------+
| conv2d_1 (Conv2D)                     | (None, 14, 14, 128)          |           73,856 |
+---------------------------------------+------------------------------+------------------+
| conv2d_2 (Conv2D)                     | (None, 14, 14, 128)          |          147,584 |
+---------------------------------------+------------------------------+------------------+
| max_pooling2d_1 (MaxPooling2D)        | (None, 7, 7, 128)            |                0 |
+---------------------------------------+------------------------------+------------------+
| flatten (Flatten)                     | (None, 6272)                 |                0 |
+---------------------------------------+------------------------------+------------------+
| dense (Dense)                         | (None, 64)                   |          401,472 |
+---------------------------------------+------------------------------+------------------+
| dropout (Dropout)                     | (None, 64)                   |                0 |
+---------------------------------------+------------------------------+------------------+
| dense_1 (Dense)                       | (None, 10)                   |              650 |
+---------------------------------------+------------------------------+------------------+
```

Training Results:

- Training Accuracy: 0.8587
- Validation Accuracy: 0.8690
- Validation Set Accuracy: 0.86

Confusion Matrix:

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 741 |   4 |  18 |  49 |   8 |   0 | 183 |   0 |   6 |   0 |
|   1 | 996 |   4 |  12 |   1 |   0 |   5 |   0 |   0 |   0 |
|   8 |   1 | 838 |  10 |  82 |   0 |  41 |   0 |   0 |   0 |
|  13 |  28 |  12 | 860 |  37 |   0 |  30 |   0 |   2 |   0 |
|   2 |   3 | 161 |  63 | 691 |   0 |  73 |   0 |   0 |   0 |
|   0 |   0 |   0 |   0 |   0 | 981 |   0 |  28 |   3 |   5 |
|  81 |   7 | 164 |  32 |  87 |   0 | 615 |   0 |  10 |   0 |
|   0 |   0 |   0 |   0 |   0 |  37 |   0 | 934 |   3 |  23 |
|   3 |   0 |   8 |   6 |  10 |   5 |   9 |   0 | 948 |   1 |
|   0 |   0 |   0 |   1 |   0 |  13 |   0 |  47 |   0 | 956 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

Learning Curves: There are multiple learning curves because I used the `cnn.bash` that was in the starter code to train it so what it would do is grab the first 10,000 data and train and then the next batch and retrain and so on until the validation batch (that gave us the confusion matrix).

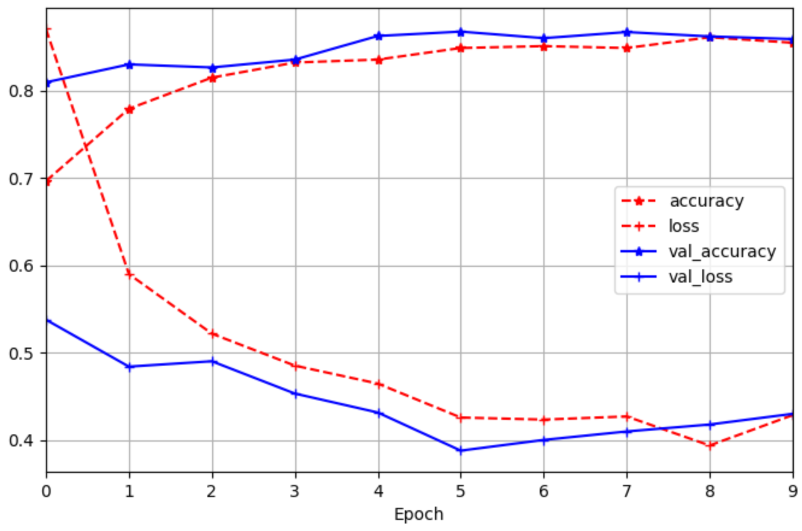


Figure 1: Learning Curve for Model A. First training.

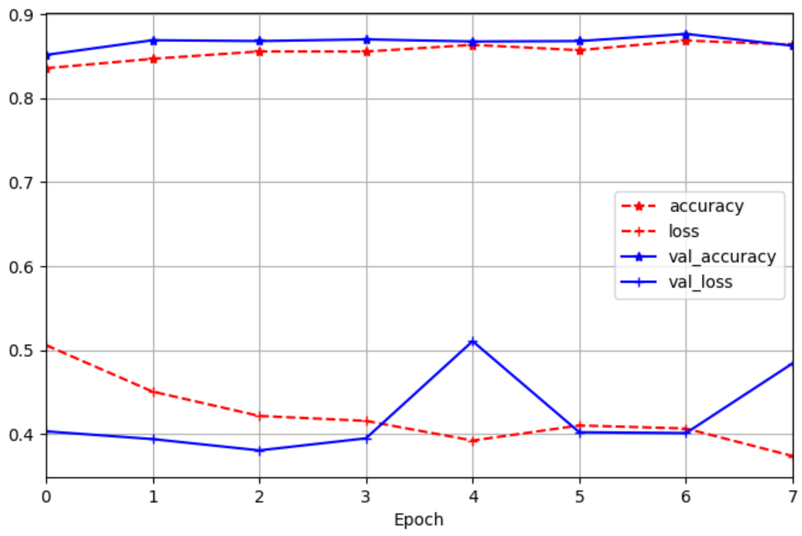


Figure 2: Learning Curve for Model A. Second training.

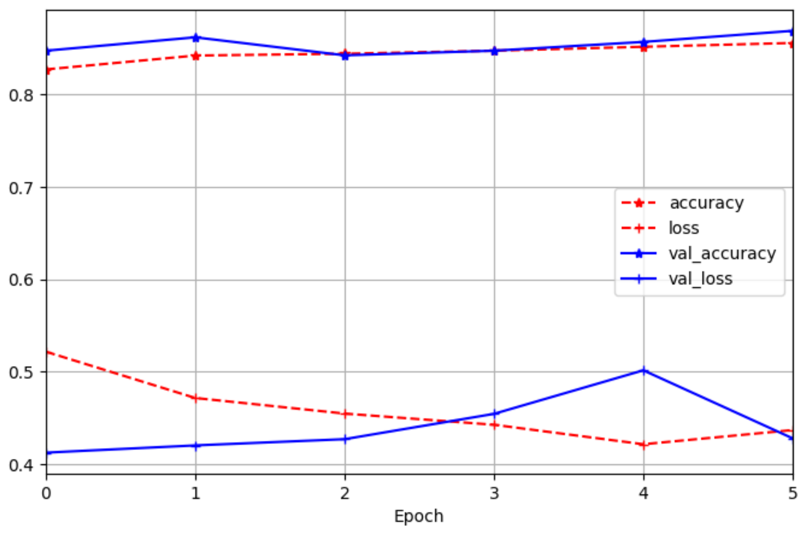


Figure 3: Learning Curve for Model A. Third training.

This model took around 15 minutes to train on all the batches and the accuracy was really good for the first try. However, as I mentioned before, this is just a test model on the Fasion MNIST to make sure everything works and take a look at how a good training epoch looks like and the graphs.

4.2 Model B

For this model I didn't change anything but the dataset from Fasion to CIFAR-10. I kept the same layer format as the last model to see how it would do with this dataset. The optimizer is `Adam(learning_rate=0.0001)`

```
+---------------------------------------+------------------------------+------------------+
| Layer (type)                          | Output Shape                 |          Param # |
+=======================================+==============================+==================+
| conv2d (Conv2D)                       | (None, 28, 28, 64)           |            9,472 |
+---------------------------------------+------------------------------+------------------+
| max_pooling2d (MaxPooling2D)          | (None, 14, 14, 64)           |                0 |
+---------------------------------------+------------------------------+------------------+
| conv2d_1 (Conv2D)                     | (None, 14, 14, 128)          |           73,856 |
+---------------------------------------+------------------------------+------------------+
| conv2d_2 (Conv2D)                     | (None, 14, 14, 128)          |          147,584 |
+---------------------------------------+------------------------------+------------------+
| max_pooling2d_1 (MaxPooling2D)        | (None, 7, 7, 128)            |                0 |
+---------------------------------------+------------------------------+------------------+
| flatten (Flatten)                     | (None, 6272)                 |                0 |
+---------------------------------------+------------------------------+------------------+
| dense (Dense)                         | (None, 64)                   |          524,352 |
+---------------------------------------+------------------------------+------------------+
| dropout (Dropout)                     | (None, 64)                   |                0 |
+---------------------------------------+------------------------------+------------------+
| dense_1 (Dense)                       | (None, 10)                   |              650 |
+---------------------------------------+------------------------------+------------------+
```

Training Results:

- Training Accuracy: 0.8197
- Validation Accuracy: 0.6495
- Validation Set Accuracy: 0.66

Confusion Matrix:

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 769 |  20 |  28 |  22 |  13 |   9 |   5 |  12 |  81 |  51 |
|  36 | 787 |   1 |   8 |   5 |   1 |   9 |   2 |  32 | 108 |
| 124 |   9 | 436 |  84 | 139 |  75 |  63 |  34 |  16 |  13 |
|  24 |   9 |  48 | 492 |  83 | 174 |  58 |  37 |  13 |  29 |
|  45 |   2 |  68 |  83 | 659 |  26 |  30 |  74 |  14 |  10 |
|   9 |   3 |  52 | 235 |  62 | 546 |  23 |  58 |   7 |   9 |
|   7 |  11 |  42 | 109 |  84 |  24 | 711 |  13 |  13 |  19 |
|  32 |   1 |  17 |  56 | 106 |  78 |   2 | 726 |   2 |  26 |
| 107 |  26 |   4 |  21 |   4 |   4 |   5 |   8 | 801 |  24 |
|  38 | 125 |   1 |  21 |   4 |   9 |   8 |  12 |  27 | 698 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

Learning Curves: There are multiple learning curves because I kept the `cnn.bash` format of training on the first batch and then the next one and so forth, but since the new dataset has 50,000 training images and we use 40,000 to train it generated a learning curve per training session.

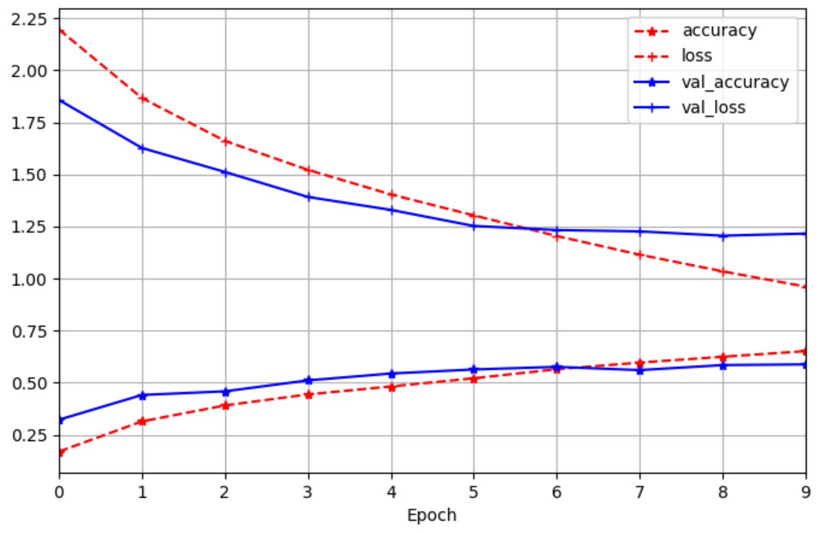


Figure 4: Learning Curve for Model B. First training.

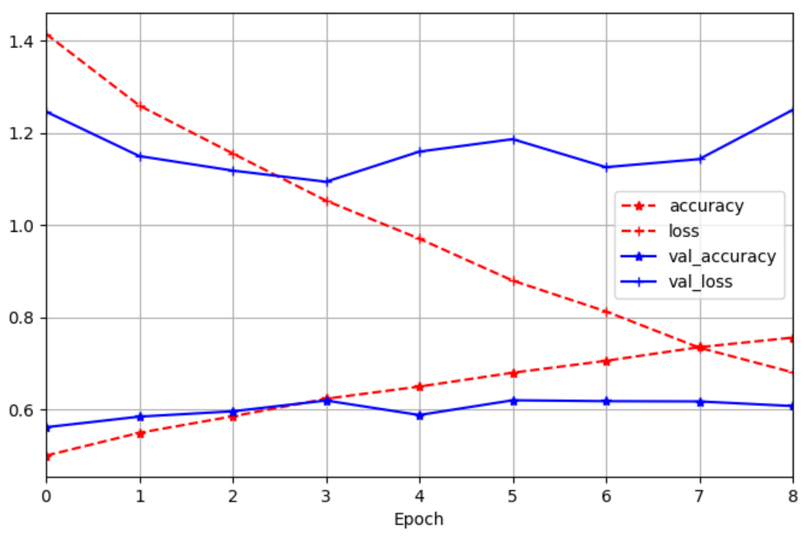


Figure 5: Learning Curve for Model B. Second training.

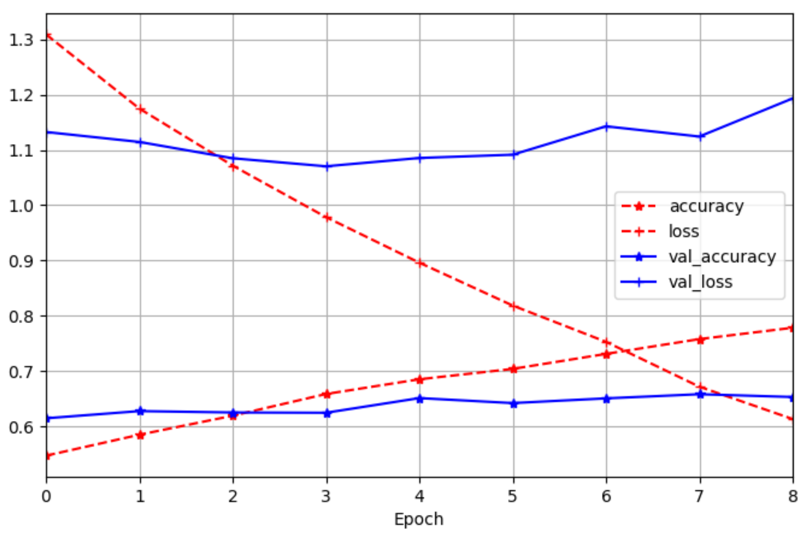


Figure 6: Learning Curve for Model B. Third training.

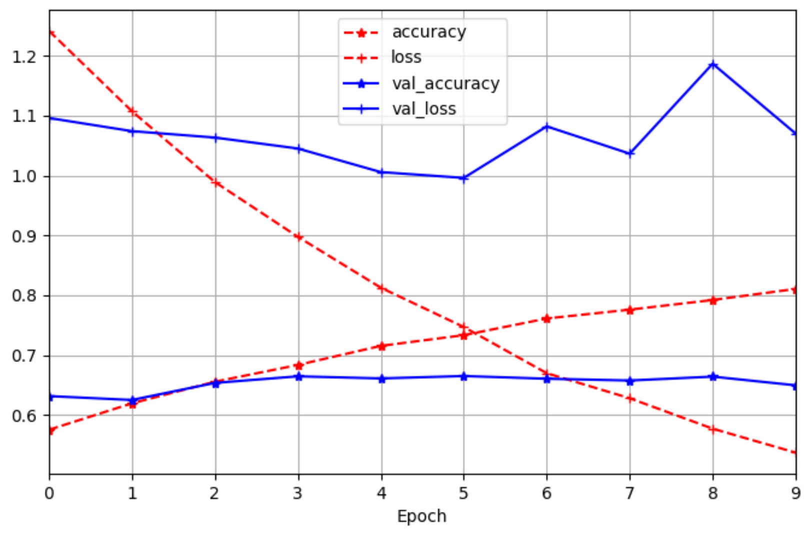


Figure 7: Learning Curve for Model B. Third training.

This model took about 30 minutes. I can see that this model was not good at generalizing and it was overfitting because towards the end the difference in between the accuracy and the validation accuracy was big (0.8197 vs 0.6495). I also realized that in the first training batch it was doing good but it stopped because I had a maximum of 10 epochs so that is something I will look into in the next models. Not great but not a bad start.

4.3 Model C

I figured out from last model that the batch size was 1 and that was slowing down the process so I changed it to 32. The optimizer is `Adam(learning_rate=0.0001)` For the structure of the model I followed the simplest in-class recommendation:

```
- Conv (bigger kernel size)
  Bigger kernel size to capture large scale info before pooling.
- Pooling
-----------
- Conv (smaller kernel size)
- Conv (smaller kernel size)
  Now we want to pay more attention to small collections of things
  so thats why smaller kernel size.
- Pooling
-----------
^ as many as wanted
-----------
- Flatten
- Dense
-----------
^ as many as wanted
- Output
```

```
+--------------------------------------+-----------------------------+-----------------+
| Layer (type)                         | Output Shape                |         Param # |
+======================================+=============================+=================+
| conv2d (Conv2D)                      | (None, 32, 32, 64)          |           9,472 |
+--------------------------------------+-----------------------------+-----------------+
| max_pooling2d (MaxPooling2D)         | (None, 16, 16, 64)          |               0 |
+--------------------------------------+-----------------------------+-----------------+
| conv2d_1 (Conv2D)                    | (None, 16, 16, 128)         |         204,928 |
+--------------------------------------+-----------------------------+-----------------+
| conv2d_2 (Conv2D)                    | (None, 16, 16, 128)         |         147,584 |
+--------------------------------------+-----------------------------+-----------------+
| max_pooling2d_1 (MaxPooling2D)       | (None, 8, 8, 128)           |               0 |
+--------------------------------------+-----------------------------+-----------------+
| conv2d_3 (Conv2D)                    | (None, 8, 8, 256)           |         295,168 |
+--------------------------------------+-----------------------------+-----------------+
| max_pooling2d_2 (MaxPooling2D)       | (None, 4, 4, 256)           |               0 |
+--------------------------------------+-----------------------------+-----------------+
| flatten (Flatten)                    | (None, 4096)                |               0 |
+--------------------------------------+-----------------------------+-----------------+
| dense (Dense)                        | (None, 128)                 |         524,416 |
+--------------------------------------+-----------------------------+-----------------+
| dropout (Dropout)                    | (None, 128)                 |               0 |
+--------------------------------------+-----------------------------+-----------------+
| dense_1 (Dense)                      | (None, 10)                  |           1,290 |
+--------------------------------------+-----------------------------+-----------------+
```

Training Results:

- Training Accuracy: 0.8036
- Validation Accuracy: 0.6620
- Validation Set Accuracy: 0.66

Confusion Matrix:

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 754 |  30 |  53 |   6 |  22 |   8 |  18 |  20 |  69 |  30 |
|  41 | 818 |   6 |   6 |   4 |   2 |  15 |   4 |  17 |  76 |
|  87 |   9 | 563 |  54 |  95 |  42 |  72 |  37 |  19 |  15 |
|  31 |  11 |  90 | 436 |  76 | 145 |  94 |  41 |  11 |  32 |
|  36 |   7 | 117 |  52 | 596 |  26 |  57 | 108 |   8 |   4 |
|   7 |   6 | 101 | 220 |  59 | 476 |  45 |  76 |   7 |   7 |
|  10 |  14 |  53 |  52 |  66 |  11 | 793 |   9 |   8 |  17 |
|  18 |   4 |  46 |  39 |  91 |  51 |   5 | 778 |   1 |  13 |
| 105 |  66 |  15 |  18 |  13 |   2 |  11 |   3 | 746 |  25 |
|  46 | 155 |  11 |   9 |   4 |   9 |  15 |  30 |  28 | 636 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

Learning Curves:

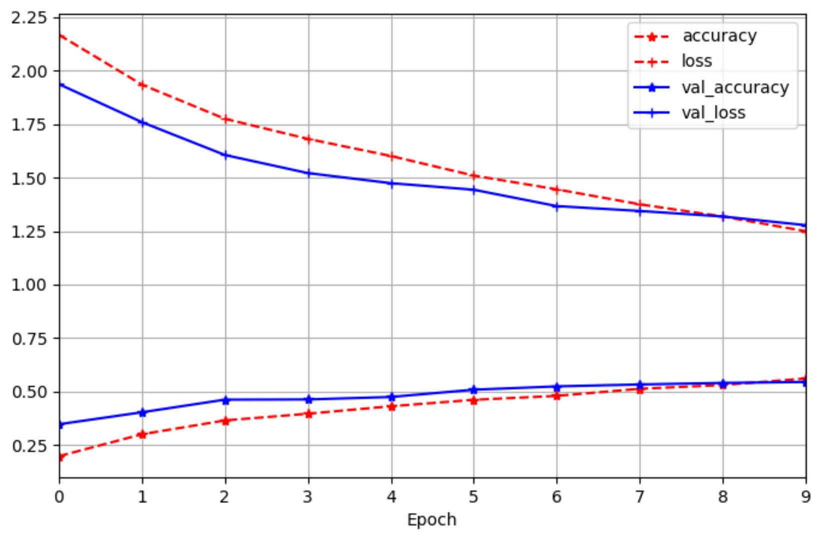


Figure 8: Learning Curve for Model C. First training.

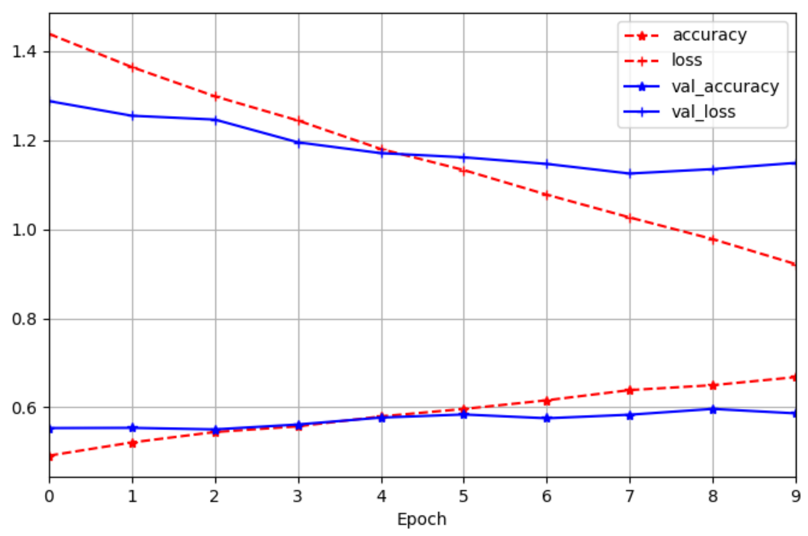


Figure 9: Learning Curve for Model C. Second training.

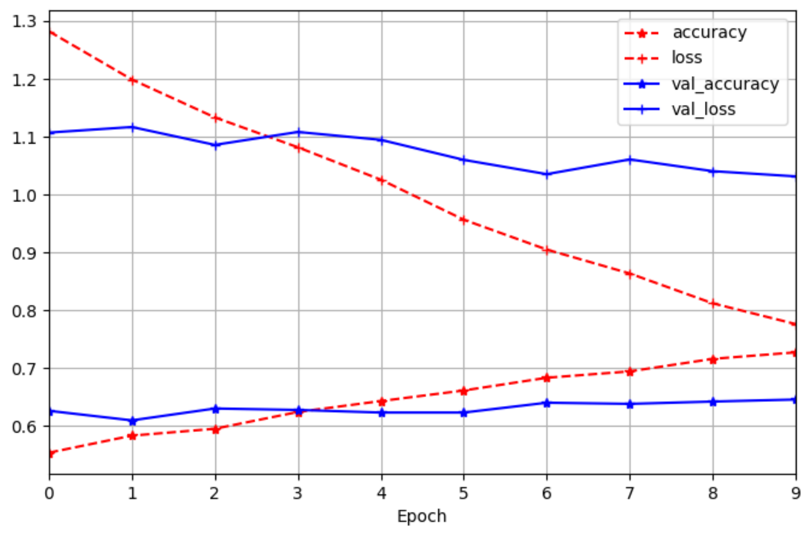


Figure 10: Learning Curve for Model C. Third training.

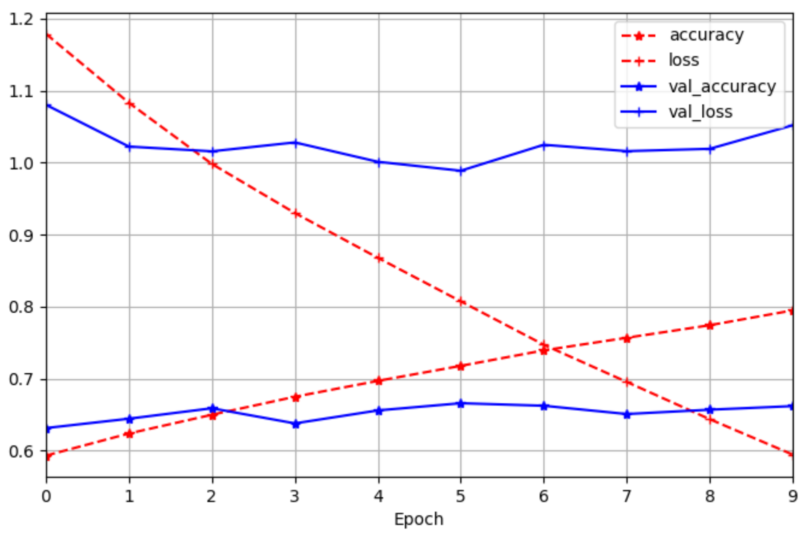
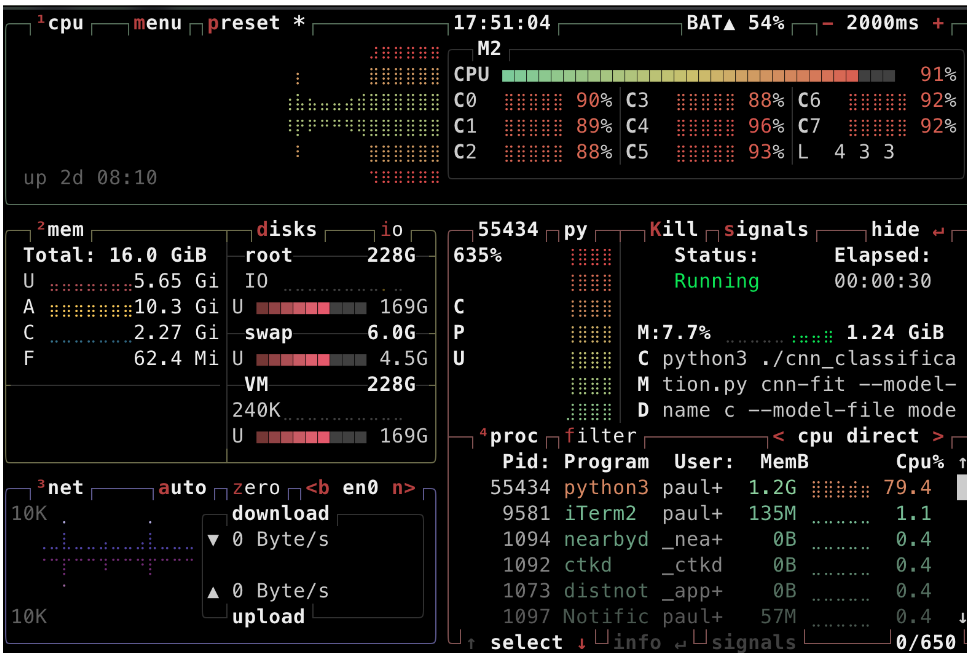


Figure 11: Learning Curve for Model C. Third training.

With the new batch size this took 20 minutes. However, my computer was boiling and it does not have fans (See picture bellow) so I decided to try and install the GPU part of Tensorflow so that it can use my mac's GPU and hope that that will make it faster and not as hot. I hadn't changed the epochs from 10 and I can still see in the first training that it would've gone for longer. The accuracy is about the same so I need to change something more.



```
 ¹cpu   menu  preset *                 17:51:04              BAT▲ 54%   - 2000ms +
                                         M2
                                         CPU                                    91%
                                         C0        90% C3        88% C6        92%
                                         C1        89% C4        96% C7        92%
                                         C2        88% C5        93% L   4 3 3
 up 2d 08:10

 ²mem                  disks      io     55434  py       Kill  signals     hide ↵
 Total: 16.0 GiB       root        228G  635%               Status:       Elapsed:
 U          5.65 Gi    IO                                   Running       00:00:30
 A          10.3 Gi   U            169G  C
 C          2.27 Gi    swap        6.0G  P               M:7.7%            1.24 GiB
 F          62.4 Mi   U            4.5G  U               C python3 ./cnn_classifica
                       VM          228G                  M tion.py cnn-fit --model-
                      240K                               D name c --model-file mode
                      U            169G    ⁴proc  filter             < cpu direct >
                                            Pid: Program  User:   MemB          Cpu% ↑
 ³net          auto  zero  <b en0 n>      55434 python3 paul+   1.2G          79.4
 10K                  download             9581 iTerm2  paul+   135M           1.1
                    ▼ 0 Byte/s             1094 nearbyd _nea+     0B           0.4
                                           1092 ctkd    _ctkd     0B           0.4
                    ▲ 0 Byte/s             1073 distnot _app+     0B           0.4
 10K                  upload               1097 Notific paul+    57M           0.4 ↓
                                         ↑ select ↓  info ↵  signals           0/650
```

Figure 12: CPU usage with btop

4.4 Model D

For this model's structure I took out the additional convolutional and pooling layers for more to see if they were important. I also figured out the GPU with TensorFlow The optimizer is `Adam(learning_rate=0.0001)`

```
+---------------------------------------+-------------------------------+------------------+
| Layer (type)                          | Output Shape                  |          Param # |
+=======================================+===============================+==================+
| conv2d (Conv2D)                       | (None, 32, 32, 64)            |            9,472 |
+---------------------------------------+-------------------------------+------------------+
| max_pooling2d (MaxPooling2D)          | (None, 16, 16, 64)            |                0 |
+---------------------------------------+-------------------------------+------------------+
| conv2d_1 (Conv2D)                     | (None, 16, 16, 128)           |          204,928 |
+---------------------------------------+-------------------------------+------------------+
| conv2d_2 (Conv2D)                     | (None, 16, 16, 128)           |          147,584 |
+---------------------------------------+-------------------------------+------------------+
| max_pooling2d_1 (MaxPooling2D)        | (None, 8, 8, 128)             |                0 |
+---------------------------------------+-------------------------------+------------------+
| flatten (Flatten)                     | (None, 8192)                  |                0 |
+---------------------------------------+-------------------------------+------------------+
| dense (Dense)                         | (None, 128)                   |        1,048,704 |
+---------------------------------------+-------------------------------+------------------+
| dense_1 (Dense)                       | (None, 128)                   |           16,512 |
+---------------------------------------+-------------------------------+------------------+
| dropout (Dropout)                     | (None, 128)                   |                0 |
+---------------------------------------+-------------------------------+------------------+
| dense_2 (Dense)                       | (None, 10)                    |            1,290 |
+---------------------------------------+-------------------------------+------------------+
```

Training Results:

- Training Accuracy: 0.2587
- Validation Accuracy: 0.4170
- Validation Set Accuracy: 0.34

Confusion Matrix:

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 406 |  26 |  43 |   5 |  14 |  18 |  71 | 130 | 131 | 166 |
|  22 | 166 |   2 |   5 |   2 |   6 | 112 |  65 |  26 | 583 |
|  93 |   4 |  77 |   7 |  53 |  68 | 399 | 198 |  29 |  65 |
|  16 |   4 |  28 |  23 |   5 | 155 | 390 | 245 |   8 |  93 |
|  36 |   3 |  40 |   5 |  53 |  30 | 515 | 266 |  14 |  49 |
|   9 |   1 |  23 |  20 |   5 | 259 | 377 | 254 |   3 |  53 |
|   6 |   7 |  19 |   1 |  10 |  17 | 772 | 129 |   8 |  64 |
|  21 |  11 |   8 |  10 |   9 |  43 | 220 | 621 |   6 |  97 |
| 189 |  33 |  26 |  14 |   3 |  35 |  57 |  39 | 321 | 287 |
|  21 |  21 |   2 |   6 |   2 |   9 |  69 |  73 |  11 | 729 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

Learning Curves:

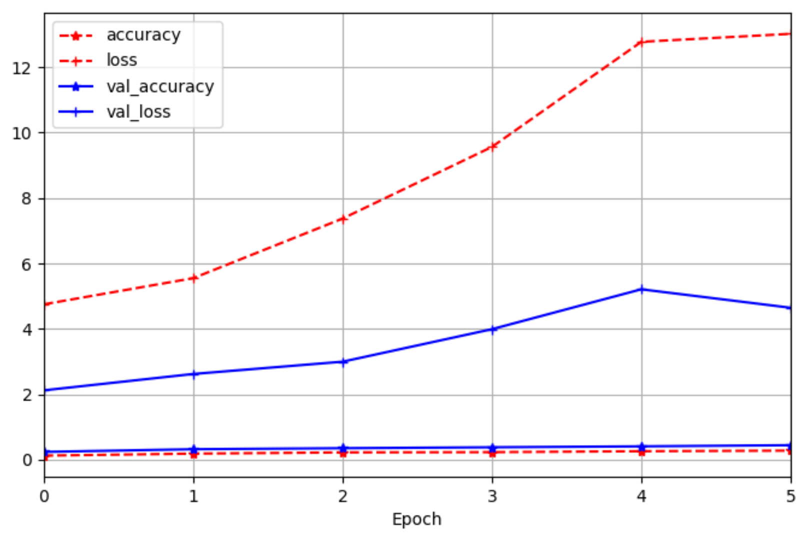


Figure 13: Learning Curve for Model D. First training.

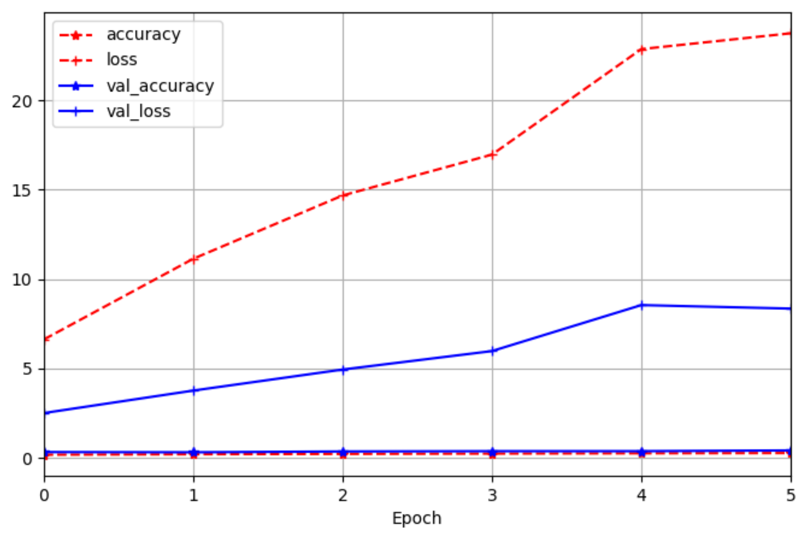


Figure 14: Learning Curve for Model D. Second training.

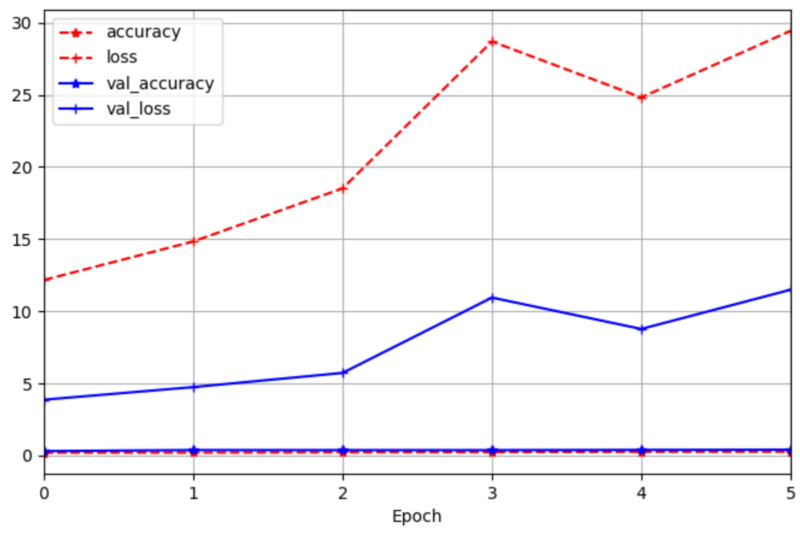


Figure 15: Learning Curve for Model D. Third training.

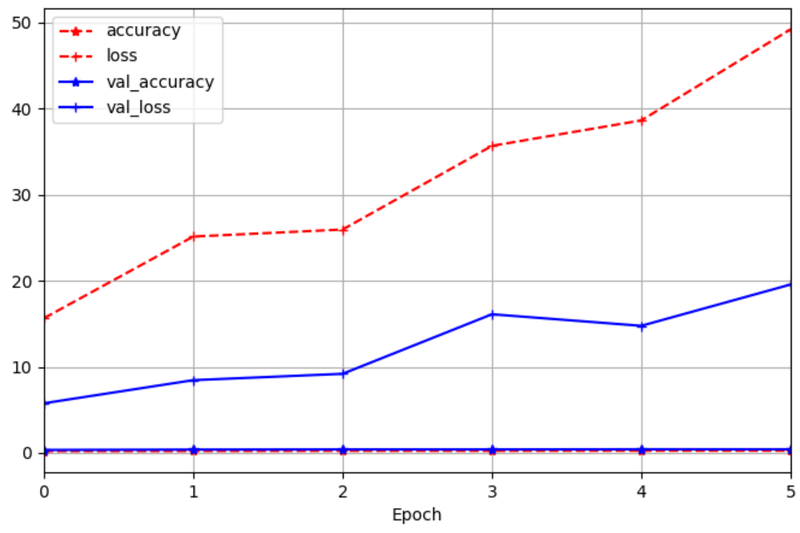


Figure 16: Learning Curve for Model D. Third training.

This one took about 6 minutes to finish training. I am glad I now have the GPU available because I would've been disappointing to wait all the time and heat for such a bad model. Now I know that the recommended structure is recommended for a reason and I will stick to it.

4.5 Model E

For this model I moved into training with all the data all at once instead of the different batches. I also changed the batch size to 16 and I added more dropout layers with smaller rates (0.25 or 0.15 instead of 0.5), I also added one conv and one pooling more and I changed the optimizer to `Adamax(learning_rate=0.001, weight_decay=1e-4)`. I also trained with 20 epochs to make sure it could converge.

```
+---------------------------------------+-----------------------------+-----------------+
| Layer (type)                          | Output Shape                |         Param # |
+=======================================+=============================+=================+
| conv2d (Conv2D)                       | (None, 32, 32, 64)          |           9,472 |
+---------------------------------------+-----------------------------+-----------------+
| max_pooling2d (MaxPooling2D)          | (None, 16, 16, 64)          |               0 |
+---------------------------------------+-----------------------------+-----------------+
| conv2d_1 (Conv2D)                     | (None, 16, 16, 128)         |         204,928 |
+---------------------------------------+-----------------------------+-----------------+
| conv2d_2 (Conv2D)                     | (None, 16, 16, 128)         |         147,584 |
+---------------------------------------+-----------------------------+-----------------+
| max_pooling2d_1 (MaxPooling2D)        | (None, 8, 8, 128)           |               0 |
+---------------------------------------+-----------------------------+-----------------+
| dropout (Dropout)                     | (None, 8, 8, 128)           |               0 |
+---------------------------------------+-----------------------------+-----------------+
| conv2d_3 (Conv2D)                     | (None, 8, 8, 256)           |         295,168 |
+---------------------------------------+-----------------------------+-----------------+
| max_pooling2d_2 (MaxPooling2D)        | (None, 4, 4, 256)           |               0 |
+---------------------------------------+-----------------------------+-----------------+
| conv2d_4 (Conv2D)                     | (None, 4, 4, 256)           |         590,080 |
+---------------------------------------+-----------------------------+-----------------+
| max_pooling2d_3 (MaxPooling2D)        | (None, 2, 2, 256)           |               0 |
+---------------------------------------+-----------------------------+-----------------+
| dropout_1 (Dropout)                   | (None, 2, 2, 256)           |               0 |
+---------------------------------------+-----------------------------+-----------------+
| flatten (Flatten)                     | (None, 1024)                |               0 |
+---------------------------------------+-----------------------------+-----------------+
| dense (Dense)                         | (None, 128)                 |         131,200 |
+---------------------------------------+-----------------------------+-----------------+
| dropout_2 (Dropout)                   | (None, 128)                 |               0 |
+---------------------------------------+-----------------------------+-----------------+
| dense_1 (Dense)                       | (None, 10)                  |           1,290 |
+---------------------------------------+-----------------------------+-----------------+
```

Training Results:

- Training Accuracy: 0.7707
- Validation Accuracy: 0.7402
- Validation Set Accuracy: 0.73

Confusion Matrix:

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 744 |  14 |  70 |  18 |  22 |  10 |   8 |  10 |  62 |  52 |
|  16 | 796 |   6 |   7 |   7 |   4 |   9 |   4 |  15 | 125 |
|  59 |   2 | 685 |  56 |  73 |  42 |  41 |  12 |  11 |  12 |
|  17 |   2 |  91 | 540 |  55 | 163 |  56 |  19 |   1 |  23 |
|  24 |   1 | 131 |  66 | 666 |  38 |  26 |  45 |   5 |   9 |
|   3 |   0 |  96 | 177 |  45 | 617 |  24 |  35 |   1 |   6 |
|   4 |   2 |  78 |  81 |  38 |  32 | 786 |   4 |   4 |   4 |
|  10 |   0 |  61 |  55 |  77 |  60 |   3 | 765 |   2 |  13 |
|  45 |  32 |  16 |  22 |   9 |   7 |   8 |   6 | 830 |  29 |
|  11 |  41 |  10 |  18 |   2 |   9 |   6 |   4 |  17 | 825 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

Learning Curve: Since I only trained once with all the data we only got one learning curve.

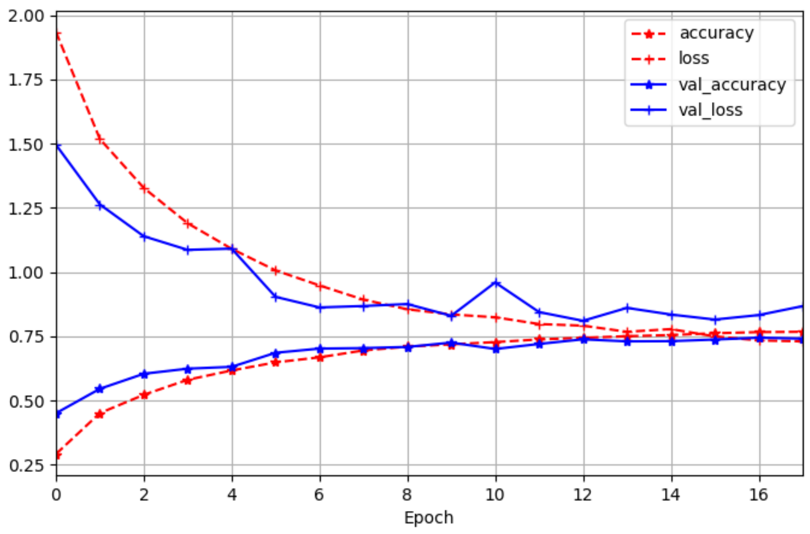


Figure 17: Learning Curve for Model E. First training.

This model took about 20 minutes to finish

4.6 Model F

I kept the same optimizer `Adamax(learning_rate=0.001, weight_decay=1e-4)` and I also remembered from last assignments to use batch normalization and activation layers so I added those all over to see if the model would generalize better. I also found `spatial_dropout2d` by asking chatgpt how to make my model better with keras, it sounded interesting so I added it.

```
+---------------------------------------+------------------------------+-----------------+
| Layer (type)                          | Output Shape                 |         Param # |
+=======================================+==============================+=================+
| conv2d (Conv2D)                       | (None, 32, 32, 64)           |           4,864 |
+---------------------------------------+------------------------------+-----------------+
| batch_normalization                   | (None, 32, 32, 64)           |             256 |
| (BatchNormalization)                  |                              |                 |
+---------------------------------------+------------------------------+-----------------+
| activation (Activation)               | (None, 32, 32, 64)           |               0 |
+---------------------------------------+------------------------------+-----------------+
| max_pooling2d (MaxPooling2D)          | (None, 16, 16, 64)           |               0 |
+---------------------------------------+------------------------------+-----------------+
| conv2d_1 (Conv2D)                     | (None, 16, 16, 128)          |         204,928 |
+---------------------------------------+------------------------------+-----------------+
| batch_normalization_1                 | (None, 16, 16, 128)          |             512 |
| (BatchNormalization)                  |                              |                 |
+--------------------------------------+------------------------------+------------------+
| activation_1 (Activation)            | (None, 16, 16, 128)          |                0 |
+--------------------------------------+------------------------------+------------------+
| conv2d_2 (Conv2D)                    | (None, 16, 16, 128)          |          147,584 |
+--------------------------------------+------------------------------+------------------+
| batch_normalization_2                | (None, 16, 16, 128)          |              512 |
| (BatchNormalization)                 |                              |                  |
+--------------------------------------+------------------------------+------------------+
| activation_2 (Activation)            | (None, 16, 16, 128)          |                0 |
+--------------------------------------+------------------------------+------------------+
| max_pooling2d_1 (MaxPooling2D)       | (None, 8, 8, 128)            |                0 |
+--------------------------------------+------------------------------+------------------+
| spatial_dropout2d (SpatialDropout2D) | (None, 8, 8, 128)            |                0 |
+--------------------------------------+------------------------------+------------------+
| conv2d_3 (Conv2D)                    | (None, 8, 8, 256)            |          295,168 |
+--------------------------------------+------------------------------+------------------+
| batch_normalization_3                | (None, 8, 8, 256)            |            1,024 |
| (BatchNormalization)                 |                              |                  |
+--------------------------------------+------------------------------+------------------+
| activation_3 (Activation)            | (None, 8, 8, 256)            |                0 |
+--------------------------------------+------------------------------+------------------+
| max_pooling2d_2 (MaxPooling2D)       | (None, 4, 4, 256)            |                0 |
+--------------------------------------+------------------------------+------------------+
| flatten (Flatten)                    | (None, 4096)                 |                0 |
+--------------------------------------+------------------------------+------------------+
| dense (Dense)                        | (None, 256)                  |        1,048,832 |
+--------------------------------------+------------------------------+------------------+
| batch_normalization_4                | (None, 256)                  |            1,024 |
| (BatchNormalization)                 |                              |                  |
+--------------------------------------+------------------------------+------------------+
| activation_4 (Activation)            | (None, 256)                  |                0 |
+--------------------------------------+------------------------------+------------------+
| dropout (Dropout)                    | (None, 256)                  |                0 |
+--------------------------------------+------------------------------+------------------+
| dense_1 (Dense)                      | (None, 10)                   |            2,570 |
+--------------------------------------+------------------------------+------------------+
```

Training Results:

- Training Accuracy: 0.9138
- Validation Accuracy: 0.7979
- Validation Set Accuracy: 0.79

Confusion Matrix:

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 816 |   5 |  42 |  16 |  16 |  13 |   6 |   2 |  68 |  26 |
|  21 | 860 |   4 |   8 |   1 |   2 |   7 |   1 |  15 |  70 |
|  47 |   4 | 691 |  54 |  65 |  51 |  45 |  15 |  12 |   9 |
|  21 |   2 |  38 | 594 |  50 | 162 |  53 |  20 |  10 |  17 |
|  20 |   1 |  48 |  67 | 774 |  34 |  17 |  40 |   7 |   3 |
|   7 |   2 |  37 | 147 |  32 | 719 |  15 |  35 |   4 |   6 |
|  11 |   3 |  36 |  54 |  25 |  17 | 876 |   2 |   5 |   4 |
|   9 |   1 |  23 |  56 |  50 |  63 |   3 | 825 |   4 |  12 |
|  30 |  16 |  10 |   9 |   3 |   6 |  10 |   2 | 905 |  13 |
|  24 |  46 |   3 |   4 |   5 |   7 |   6 |   1 |  31 | 816 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

Since this is my best model I am going to predict on the testing data and see if I actually get somewhat close to the validation Set Accuracy: **Testing Confusion Matrix:**

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 820 |  11 |  50 |  16 |   9 |   5 |  10 |   6 |  48 |  25 |
|  16 | 871 |   2 |   5 |   1 |   3 |   6 |   0 |  22 |  74 |
|  47 |   2 | 674 |  60 |  70 |  54 |  47 |  27 |   9 |  10 |
|  18 |   3 |  46 | 606 |  59 | 166 |  43 |  27 |  16 |  16 |
|  18 |   2 |  48 |  52 | 780 |  30 |  29 |  29 |  10 |   2 |
|  12 |   0 |  22 | 145 |  42 | 715 |  18 |  35 |   2 |   9 |
|   6 |   3 |  24 |  51 |  26 |  25 | 855 |   1 |   4 |   5 |
|   8 |   1 |  24 |  34 |  44 |  59 |   4 | 808 |   6 |  12 |
|  47 |  12 |  10 |  11 |   6 |   9 |   5 |   3 | 884 |  13 |
|  39 |  49 |   7 |   4 |   4 |   6 |   3 |   4 |  32 | 852 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

```
              precision    recall  f1-score   support

           0       0.80      0.82      0.81      1000
           1       0.91      0.87      0.89      1000
           2       0.74      0.67      0.71      1000
           3       0.62      0.61      0.61      1000
           4       0.75      0.78      0.76      1000
           5       0.67      0.71      0.69      1000
           6       0.84      0.85      0.85      1000
           7       0.86      0.81      0.83      1000
           8       0.86      0.88      0.87      1000
           9       0.84      0.85      0.84      1000

    accuracy                           0.79     10000
   macro avg       0.79      0.79      0.79     10000
weighted avg       0.79      0.79      0.79     10000
```

5 Model Comparison

Table 1: Comparison of CNN Models (B-F) on CIFAR-10

Model	Training Acc.	Val. Acc.	Val. Set Acc.	Key Features
B	0.8197	0.6495	0.66	Same as A but on CIFAR-10, overfitting observed
C	0.8036	0.6620	0.66	Modified structure (3 Conv + 3 Pool + 1 Dense), batch size=32
D	0.2587	0.4170	0.34	Reduced layers (2 Conv + 2 Pool + 2 Dense), poor performance
E	0.7707	0.7402	0.73	Added dropout (0.25/0.15), more Conv/Pool layers, full data
F	**0.9138**	**0.7979**	**0.79**	Added BatchNorm, Spatial-Dropout, Activation layers, best generalization

6 Conclusions

The results prove that convolutional neural networks can achieve strong performance on the CIFAR-10 dataset, with my best model (Model F) reaching 79.7% validation accuracy and 79% test accuracy. Several key insights:

- **Architecture Matters**: The progression from Model B to F shows that proper layer configuration (multiple convolutional blocks with BatchNorm and dropout) significantly impacts performance. Model F's inclusion of spatial dropout and batch normalization proved particularly effective at preventing overfitting while maintaining high accuracy.
- **Training Strategy**: Using the full dataset (Model E onward) rather than batched training improved results substantially. The Adamax optimizer with weight decay ($1e^{-4}$) also showed better convergence than the initial Adam implementation.
- **Current Limitations**: While Model F performs well, there remains a noticeable gap between training (91.4%) and validation accuracy (79.8%), suggesting some persistent overfitting. The model also shows weaker performance on classes like ”cat” and ”dog”, which are visually similar.

These results confirm that careful architecture design and regularization are crucial for computer vision tasks, and that there remains room for improvement beyond the 80% accuracy threshold.