Dara Guadalupe Carrasquillo
Dylan Rodríguez Flores

01/23/2024

IronHack: Convoluted Neural Networks Project Report

I. Description of the chosen CNN architecture

Our model has the following architecture:

```
Model: "sequential_18"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                     ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ conv2d_85 (Conv2D)               │ (None, 32, 32, 32)     │           896 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_86 (Conv2D)               │ (None, 32, 32, 32)     │         9,248 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ batch_normalization_30           │ (None, 32, 32, 32)     │           128 │
│ (BatchNormalization)             │                        │               │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_46 (MaxPooling2D)  │ (None, 16, 16, 32)     │             0 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_87 (Conv2D)               │ (None, 16, 16, 64)     │        18,496 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_88 (Conv2D)               │ (None, 16, 16, 64)     │        36,928 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ batch_normalization_31           │ (None, 16, 16, 64)     │           256 │
│ (BatchNormalization)             │                        │               │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_47 (MaxPooling2D)  │ (None, 8, 8, 64)       │             0 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_89 (Conv2D)               │ (None, 8, 8, 128)      │        73,856 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_90 (Conv2D)               │ (None, 8, 8, 128)      │       147,584 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ batch_normalization_32           │ (None, 8, 8, 128)      │           512 │
│ (BatchNormalization)             │                        │               │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_48 (MaxPooling2D)  │ (None, 4, 4, 128)      │             0 │
```

```
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ flatten_18 (Flatten)            │ (None, 2048)           │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout_27 (Dropout)            │ (None, 2048)           │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_28 (Dense)                │ (None, 128)            │       262,272 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout_28 (Dropout)            │ (None, 128)            │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_29 (Dense)                │ (None, 10)             │         1,290 │
└─────────────────────────────────┴────────────────────────┴───────────────┘
```

Total params: 551,466 (2.10 MB)

Trainable params: 551,018 (2.10 MB)

Non-trainable params: 448 (1.75 KB)

II. Explanation of preprocessing steps

For this step we loaded the data and then used the following code:

```
# Normalize the pixel values to a range of [0, 1]
x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0
```

Normalization: Scales pixel values to [0, 1] to improve model performance, stability, and convergence.

```
# Convert the labels into one-hot encoding
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)
```

One-Hot Encoding: Converts integer labels into a vector format to avoid ordinal assumptions and ensure compatibility with the loss function.

III. Details of the training process (e.g., learning rate, batch size, number of epochs)

Details:

1. Model fit: The model seemed to stabilize around the 20 epochs so we left it at that number and configured early stopping to halt the training when the monitor variable did not improve further. Batch size started at 128 but we quickly figured that smaller sizes allowed for better learning and more robust results.

    a. Epochs: 20
    b. Batch size: 64
2. Early Stopping: Implemented to have the model halt all training procedures when the val_loss did not improve for three consecutive epochs.
    a. Monitor: val_loss
    b. Patience: 3
    c. Restore Best Weights: True
3. Learning Scheduler: Implemented to drop the learning rate by a factor of 0.5 to prevent overfitting in later epochs.
    a. Monitor: val_loss
    b. Factor: 0.5
    c. Patience: 2
    d. Min_lr: 0.0000001

IV. Results and analysis of model's performance

With the model displayed above we managed to achieve a test accuracy of 0.83 and a test loss of 0.58. Past this point, the model would need to increase in complexity to achieve higher results. Learning took a longer time than first stipulated, however, within the given time frame, it has achieved reasonable results for its low complexity. A confusion matrix was created in the testing, and we concluded that the pictures depicting animals seem to be giving the model much more trouble than any other category. Cats were most frequently incorrectly labeled as dogs and vice versa.

V. Best model

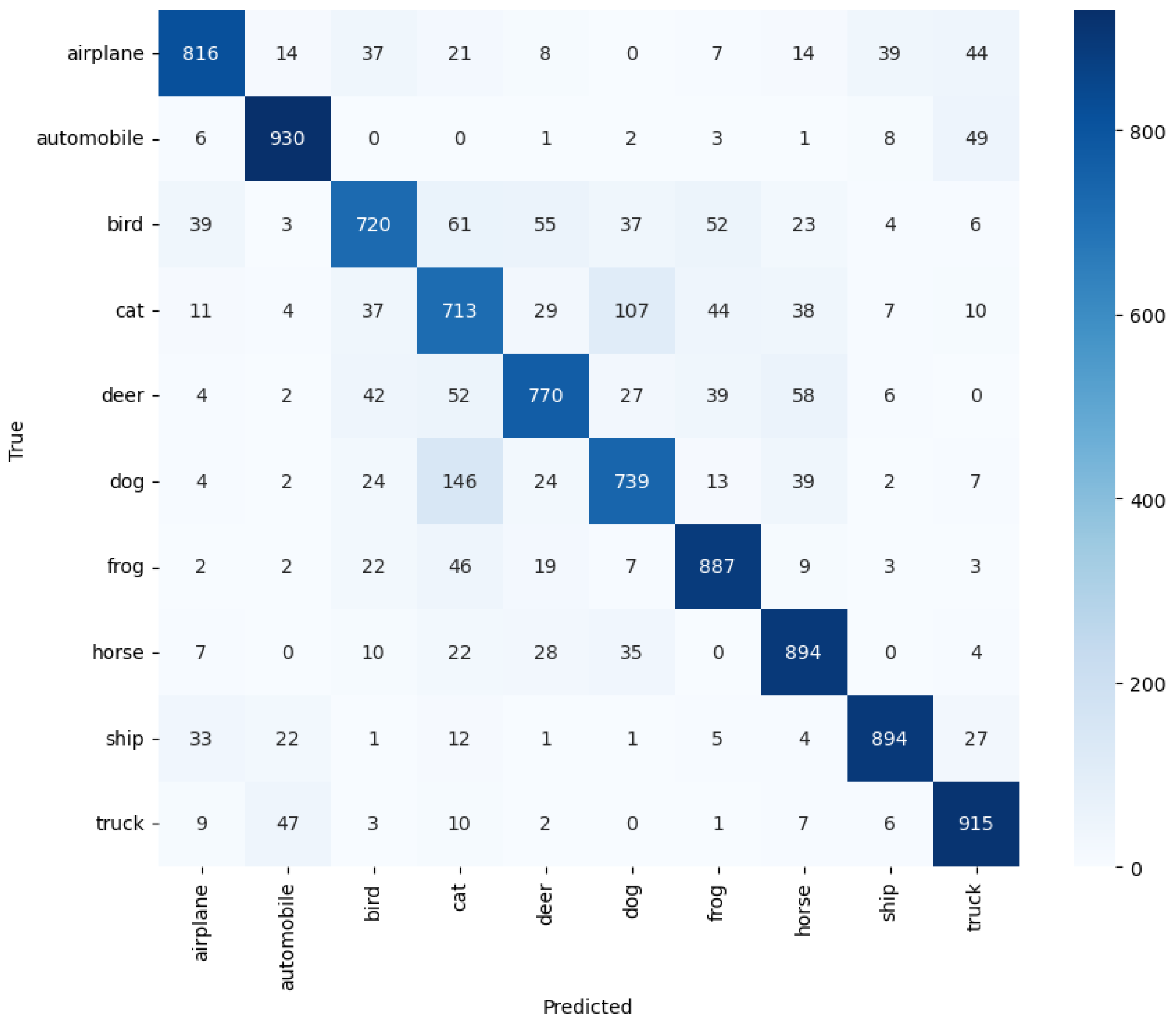
The model in this report outperformed all previous iterations, yielding a test accuracy score of 0.83 and a test loss of 0.58. The previous iterations had trouble with overfitting the data and subsequent revisions showed signs of underfitting. We knew our model would need to be somewhere in the middle and so we adjusted the layers until there was no further improvement. After that, we fine tuned the training details (epochs, batch size, etc.) to achieve higher results. This did improve the results further until we obtained the model displayed above.

VI. Insights gained from the experimentation process.

The most important insight gained from the experimentation was the importance of the architecture of the model and how the layers interact. Removing or adding a single element to one of the layers can yield dramatically different results on the testing. Furthermore, there are many ways in which the models can be fine tuned and optimized for better performance. Sometimes the most complex model is not necessarily the best one and it was very interesting to see that pacing the model and having it learn slowly yielded better results overall.

VII. Visualizations and diagrams

Confusion matrix of the test results:



True \ Predicted	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	816	14	37	21	8	0	7	14	39	44
automobile	6	930	0	0	1	2	3	1	8	49
bird	39	3	720	61	55	37	52	23	4	6
cat	11	4	37	713	29	107	44	38	7	10
deer	4	2	42	52	770	27	39	58	6	0
dog	4	2	24	146	24	739	13	39	2	7
frog	2	2	22	46	19	7	887	9	3	3
horse	7	0	10	22	28	35	0	894	0	4
ship	33	22	1	12	1	1	5	4	894	27
truck	9	47	3	10	2	0	1	7	6	915