**Assignment 2: Convolution:**

Steven Kalinoff  
3/23/2024

Adv. Machine Learning 64061

***1.Consider the Cats & Dogs example. Start initially with a training sample of 1000, a  
validation sample of 500, and a test sample of 500 (like in the text). Use any technique  
to reduce overfitting and improve performance in developing a network that you train  
from scratch. What performance did you achieve?***

The first technique I used to prevent overfitting is adding a dropout layer. Dropout will randomly kill neurons in the model and because of this the model will need to “adapt” to this change by creating new connections. At 60% dropout the accuracy was lower than the unadjusted data. The 2nd test I ran was at 30% dropout. It performed more closely to the standard model in terms of predicting accuracy but did have less validation loss. I hypothesize that this model will preform best with no dropout.

model = models.Sequential()

model.add(layers.Conv2D(32, (3, 3), activation='relu',

input\_shape=(150, 150, 3)))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64, (3, 3), activation='relu'))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(128, (3, 3), activation='relu'))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(128, (3, 3), activation='relu'))

model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Flatten())

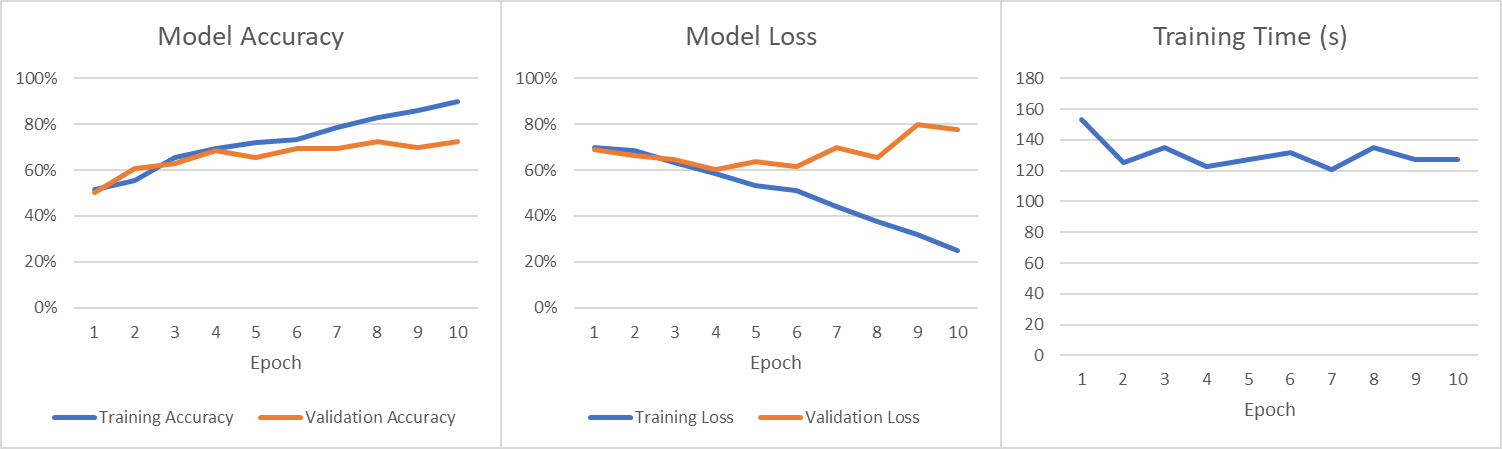
model.add(layers.Dropout(0.6)) # Added dropout here

model.add(layers.Dense(512, activation='relu'))

model.add(layers.Dense(1, activation='sigmoid'))

Results from adding dropout:

Before:



After:

Figure 1: No Dropout

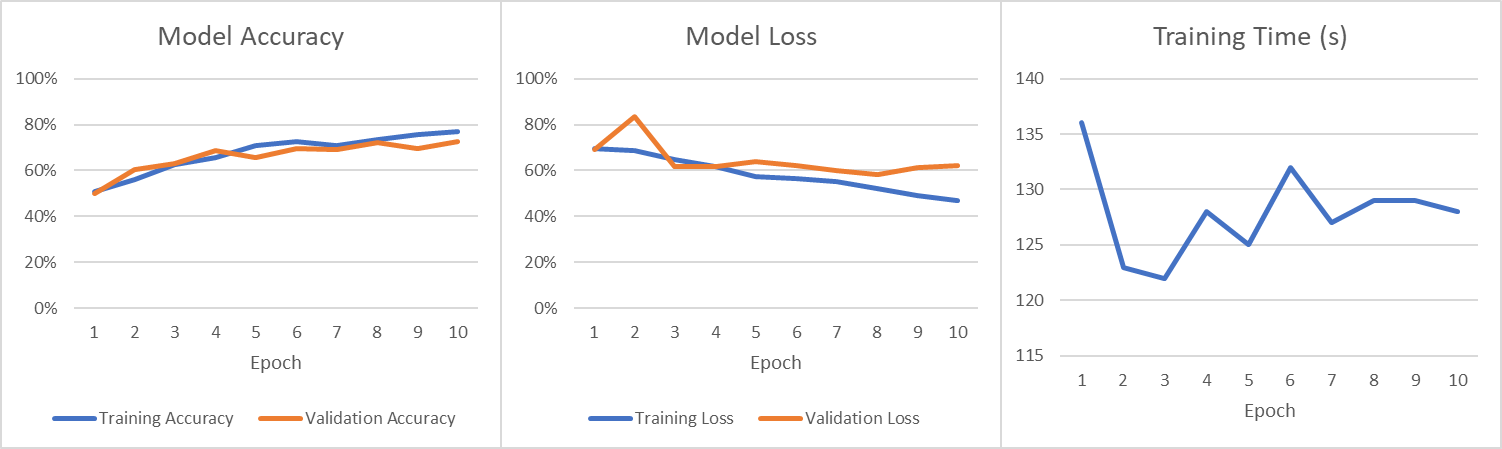


Figure 2: 60% Dropout

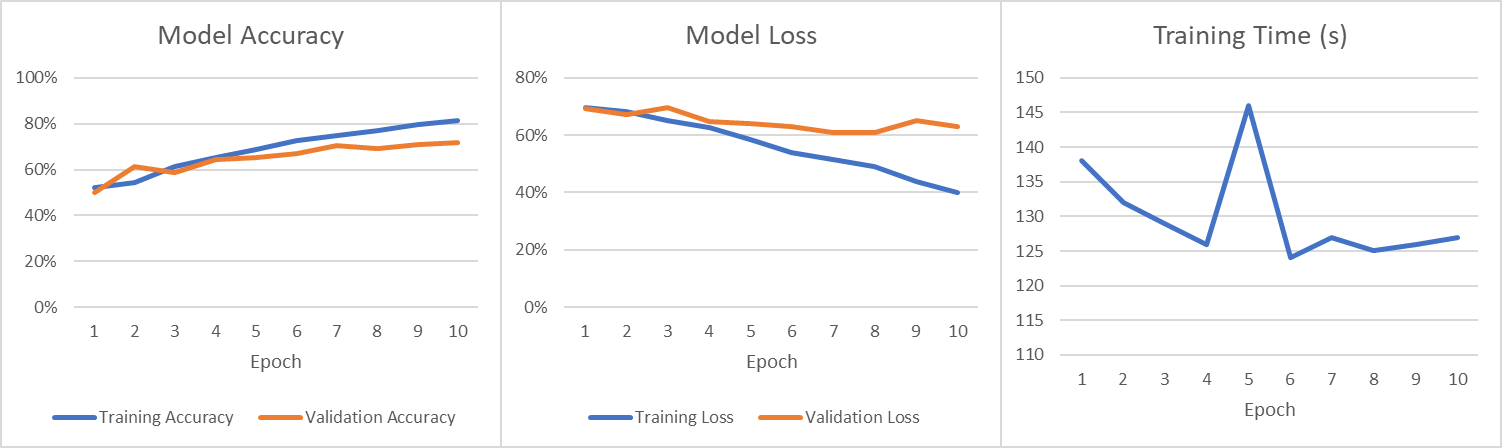


Figure 3: 30% Dropout

***2. Increase your training sample size. You may pick any amount. Keep the validation and  
test samples the same as above. Optimize your network (again training from scratch).  
What performance did you achieve?***

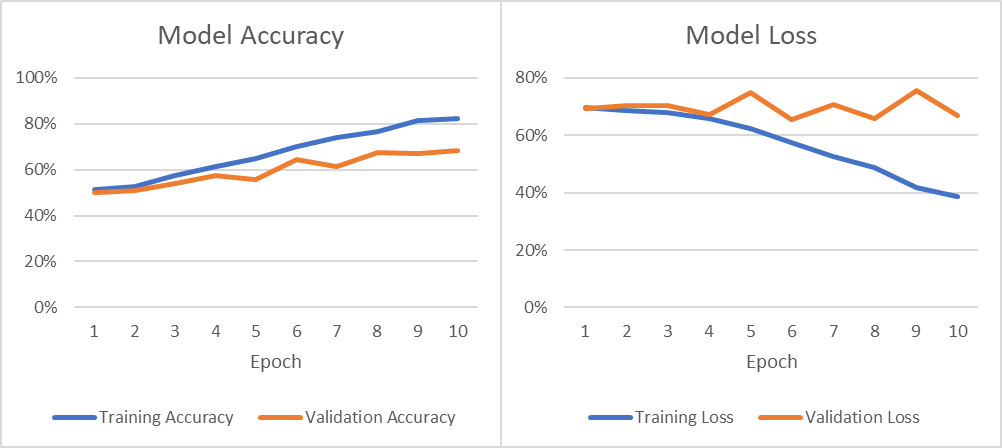
Sample size of 1250 each. In adding the 250 each or another 500 samples we achieved slightly better accuracy with  


Figure 4: + 250 training samples

***3. Now change your training sample so that you achieve better performance than those  
from Steps 1 and 2. This sample size may be larger, or smaller than those in the previous  
steps. The objective is to find the ideal training sample size to get best prediction  
results.***

Sample size of 1500 each. This uses the maximum amount of data in the smaller data set 3,000 for training and 500 for both testing and validation. This model appears to have less validation loss and higher accuracy, and this should be the most accurate model using this method of varying training data.

A graph with blue and orange lines

Description automatically generated

Figure 5: max samples

By increasing the batch size our training accuracy goes down, but our validation accuracy goes up nearing 80%.

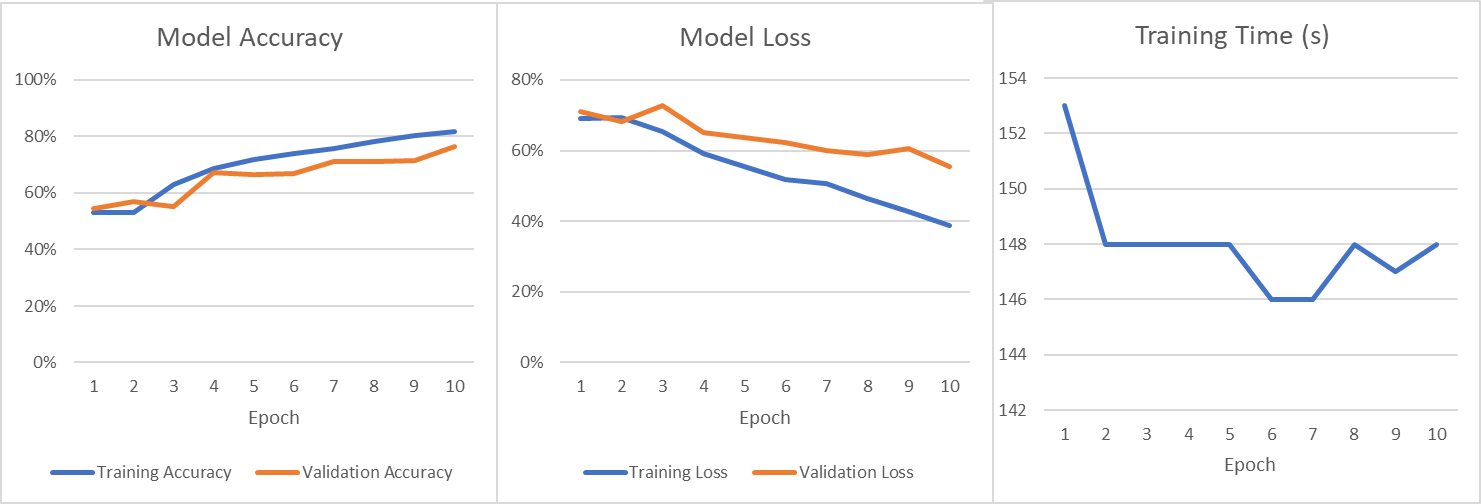


Figure 6: max samples (larger batches)

***4. Repeat Steps 1-3, but now using a pretrained network. The sample sizes you use in  
Steps 2 and 3 for the pretrained network may be the same or different from those using  
the network where you trained from scratch. Again, use any and all optimization  
techniques to get best performance.***

By using a pretrained network a few things happened inside the model. The loss was very high but quickly fell below the typical loss of previous models. Both accuracy and validation accuracy were low at the start but exceeded previous top line accuracy by the 8th epoch. This came at a huge cost of time. Most other tests concluded around 20ish minutes this final test using a pretrained network took almost an hour and a half leading to difficulties in retesting for further validation efforts.

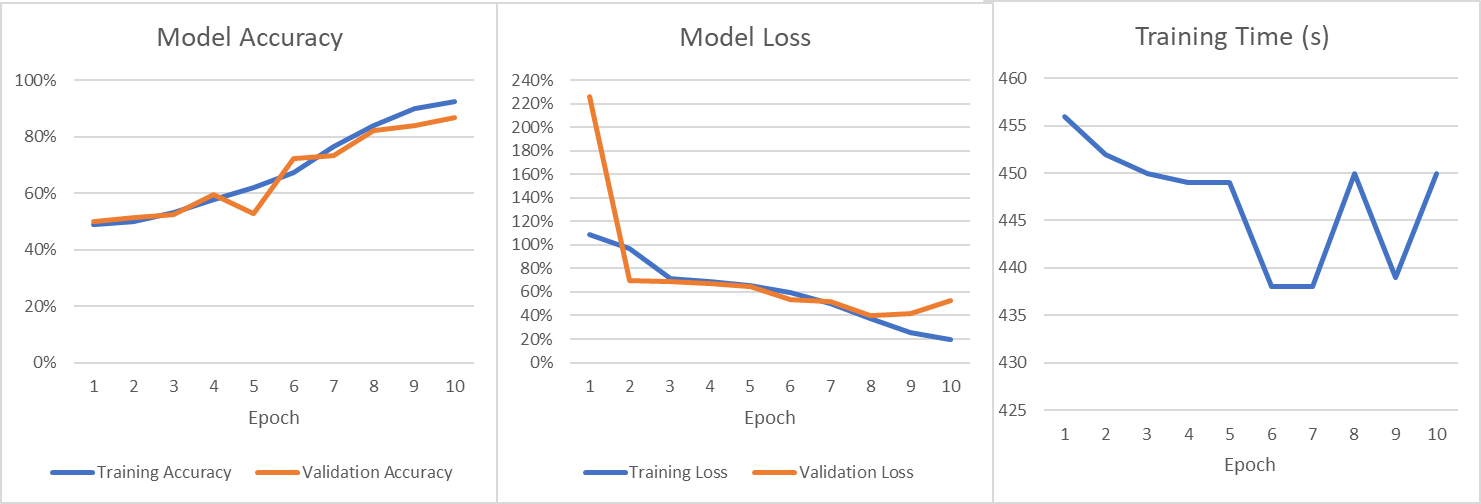


Figure 7: Pretrained

***Write a report summarizing your findings. What is the relationship between training sample  
size and choice of network?***

During this assignment our goal was to improve the validation accuracy of this model which was to classify photos of cats and dogs from random imagines provided. The imagines did have at least a cat or dog in each. This model looked for patterns from the imagines provided and could be improved by artificially creating more imagines from the existing data by manipulating the files themselves or more training data could be feed into the system. Overall, a positive correlation between the size of the training data and the validation accuracy was seen.

Dropout effectively reduced some of the feature finding in training and lead to less accurate models. Adding more training data always increased the accuracy of the training model but causes some “waves” in the validation epochs. I think that thinning the model during the later stages of the network only may have resulted in less of this effect and perhaps higher validation accuracy.

Changing the batch size to attempt in finding higher level key features did not improve accuracy and greatly increased loss and training time volatility.

Perhaps the greatest overall improvement came from using the pretrained data. This data (after the first epoch) was the most refined and was the only one to reach into the 90% range in terms of test data. The validation accuracy also improved greatly in the last 3 epochs reaching almost 87% accuracy. The trade off for this model as mentioned above is that the training time tripled, perhaps the google colab resources are not as effective at running this kind of network. This assignment shows that sample sizes must be chosen with care and experimentation. It also shows that key performance indicators can show where the model may start to overfit, and we can monitor epoch progress to shut down models that start to decline in training data accuracy.

***Below is the data corresponding to each plot in the report above.***

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Epoch** | **Training Time (s)** | **Total Time (s)** | **Steps per Second** | **Training Loss** | **Training Accuracy** | **Validation Loss** | **Validation Accuracy** |
| 1 | 153 | 153 | 2 | 0.6977 | 0.517 | 0.6914 | 0.501 |
| 2 | 125 | 278 | 2 | 0.6831 | 0.5565 | 0.6621 | 0.606 |
| 3 | 135 | 413 | 2 | 0.6329 | 0.6555 | 0.6455 | 0.628 |
| 4 | 123 | 536 | 2 | 0.5832 | 0.692 | 0.6019 | 0.686 |
| 5 | 127 | 663 | 2 | 0.5339 | 0.7215 | 0.6367 | 0.655 |
| 6 | 132 | 795 | 2 | 0.5122 | 0.735 | 0.6167 | 0.695 |
| 7 | 121 | 916 | 2 | 0.4409 | 0.7855 | 0.6981 | 0.693 |
| 8 | 135 | 1051 | 2 | 0.3754 | 0.827 | 0.6542 | 0.723 |
| 9 | 127 | 1178 | 2 | 0.3182 | 0.8575 | 0.7986 | 0.697 |
| 10 | 127 | 1305 | 2 | 0.2515 | 0.9 | 0.7762 | 0.724 |

Table 1: No Dropout

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Epoch** | **Training Time (s)** | **Total Time (s)** | **Steps per Second** | **Training Loss** | **Training Accuracy** | **Validation Loss** | **Validation Accuracy** |
| 1 | 136 | 136 | 2 | 0.697 | 0.506 | 0.6929 | 0.5 |
| 2 | 123 | 259 | 2 | 0.6854 | 0.5605 | 0.8341 | 0.502 |
| 3 | 122 | 381 | 2 | 0.6482 | 0.625 | 0.615 | 0.645 |
| 4 | 128 | 509 | 2 | 0.6168 | 0.658 | 0.6192 | 0.652 |
| 5 | 125 | 634 | 2 | 0.575 | 0.709 | 0.6375 | 0.622 |
| 6 | 132 | 766 | 2 | 0.5664 | 0.7255 | 0.6204 | 0.664 |
| 7 | 127 | 893 | 2 | 0.5521 | 0.71 | 0.5995 | 0.696 |
| 8 | 129 | 1022 | 2 | 0.5195 | 0.7365 | 0.5832 | 0.702 |
| 9 | 129 | 1151 | 2 | 0.4915 | 0.7555 | 0.6126 | 0.707 |
| 10 | 128 | 1279 | 2 | 0.4709 | 0.7675 | 0.6231 | 0.678 |

Table 2: 60% Dropout

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Epoch** | **Training Time (s)** | **Total Time (s)** | **Steps per Second** | **Training Loss** | **Training Accuracy** | **Validation Loss** | **Validation Accuracy** |
| 1 | 138 | 138 | 2 | 0.6963 | 0.523 | 0.6919 | 0.5 |
| 2 | 132 | 270 | 2 | 0.6829 | 0.545 | 0.6715 | 0.614 |
| 3 | 129 | 399 | 2 | 0.6518 | 0.6125 | 0.6978 | 0.589 |
| 4 | 126 | 525 | 2 | 0.6249 | 0.6515 | 0.6466 | 0.642 |
| 5 | 146 | 671 | 2 | 0.586 | 0.688 | 0.6408 | 0.654 |
| 6 | 124 | 795 | 2 | 0.538 | 0.728 | 0.6303 | 0.672 |
| 7 | 127 | 922 | 2 | 0.5133 | 0.747 | 0.6108 | 0.704 |
| 8 | 125 | 1047 | 2 | 0.4892 | 0.769 | 0.609 | 0.691 |
| 9 | 126 | 1173 | 2 | 0.4371 | 0.7945 | 0.6521 | 0.708 |
| 10 | 127 | 1300 | 2 | 0.4015 | 0.8125 | 0.629 | 0.719 |

Table 3: 30% Dropout

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Epoch** | **Loss** | **Accuracy** | **Val\_Loss** | **Val\_Accuracy** |
| 1 | 0.6979 | 0.5136 | 0.6931 | 0.5 |
| 2 | 0.6878 | 0.5252 | 0.703 | 0.508 |
| 3 | 0.6803 | 0.5736 | 0.7043 | 0.54 |
| 4 | 0.66 | 0.612 | 0.6738 | 0.576 |
| 5 | 0.6239 | 0.6508 | 0.7479 | 0.556 |
| 6 | 0.5746 | 0.7008 | 0.6545 | 0.646 |
| 7 | 0.5265 | 0.742 | 0.7061 | 0.616 |
| 8 | 0.4872 | 0.766 | 0.6588 | 0.674 |
| 9 | 0.4175 | 0.8124 | 0.7553 | 0.67 |
| 10 | 0.3851 | 0.8212 | 0.6687 | 0.682 |

Table 4: +250 training data

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Epoch** | **Training Time (s)** | **Total Time (s)** | **Steps per Second** | **Loss** | **Accuracy** | **Val\_Loss** | **Val\_Accuracy** |
| 1 | 151 | 151 | 2 | 0.6957 | 0.5227 | 0.6918 | 0.538 |
| 2 | 153 | 304 | 2 | 0.6643 | 0.6067 | 0.6629 | 0.594 |
| 3 | 147 | 451 | 2 | 0.6137 | 0.668 | 0.6307 | 0.656 |
| 4 | 143 | 594 | 2 | 0.569 | 0.7017 | 0.5928 | 0.688 |
| 5 | 141 | 735 | 1 | 0.5346 | 0.7283 | 0.589 | 0.714 |
| 6 | 141 | 876 | 2 | 0.4871 | 0.7613 | 0.6082 | 0.696 |
| 7 | 147 | 1023 | 2 | 0.4491 | 0.7823 | 0.5792 | 0.746 |
| 8 | 146 | 1169 | 2 | 0.3839 | 0.828 | 0.6268 | 0.71 |
| 9 | 143 | 1312 | 2 | 0.3267 | 0.8527 | 0.7278 | 0.758 |
| 10 | 152 | 1464 | 2 | 0.2616 | 0.8853 | 0.8617 | 0.736 |

Table 5: Max samples

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Epoch** | **Training Time (s)** | **Total Time (s)** | **Steps per Second** | **Loss** | **Accuracy** | **Val\_Loss** | **Val\_Accuracy** |
| 1 | 153 | 153 | 5 | 0.6915 | 0.531 | 0.7112 | 0.546 |
| 2 | 148 | 301 | 5 | 0.6932 | 0.5317 | 0.6822 | 0.568 |
| 3 | 148 | 449 | 5 | 0.6529 | 0.63 | 0.7277 | 0.552 |
| 4 | 148 | 597 | 5 | 0.591 | 0.6857 | 0.6515 | 0.672 |
| 5 | 148 | 745 | 5 | 0.5554 | 0.7187 | 0.6369 | 0.664 |
| 6 | 146 | 891 | 5 | 0.5177 | 0.7383 | 0.6227 | 0.67 |
| 7 | 146 | 1037 | 5 | 0.5054 | 0.7577 | 0.6012 | 0.712 |
| 8 | 148 | 1185 | 5 | 0.4639 | 0.7803 | 0.5886 | 0.71 |
| 9 | 147 | 1332 | 5 | 0.4272 | 0.8043 | 0.6053 | 0.716 |
| 10 | 148 | 1480 | 5 | 0.3883 | 0.816 | 0.5557 | 0.764 |

Table 6: Max Sample (larger batches)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Epoch** | **Training Time (s)** | **Total Time (s)** | **Steps per Second** | **Loss** | **Accuracy** | **Val\_Loss** | **Val\_Accuracy** |
| **1** | **456** | **456** | **47** | **1.0915** | **0.487** | **2.2585** | **0.5** |
| **2** | **452** | **908** | **47** | **0.9648** | **0.5** | **0.6929** | **0.514** |
| **3** | **450** | **1358** | **46** | **0.7159** | **0.532** | **0.6872** | **0.524** |
| **4** | **449** | **1807** | **45** | **0.6868** | **0.578** | **0.6687** | **0.596** |
| **5** | **449** | **2256** | **46** | **0.6503** | **0.62** | **0.6428** | **0.526** |
| **6** | **438** | **2694** | **45** | **0.5919** | **0.674** | **0.5381** | **0.722** |
| **7** | **438** | **3132** | **45** | **0.5011** | **0.763** | **0.5188** | **0.734** |
| **8** | **450** | **3582** | **47** | **0.371** | **0.839** | **0.3955** | **0.822** |
| **9** | **439** | **4021** | **45** | **0.2588** | **0.898** | **0.4183** | **0.838** |
| **10** | **450** | **4021** | **47** | **0.1942** | **0.9229** | **0.5249** | **0.867** |

Table 7: Pretrained