Political Meme Classification Phase 5

Kate Arendes
Introduction to Deep Learning

April 3, 2022

Contents

1 Effects of Normalization **1**
1.1 Dropout 1
1.2 Batch Normalization 3
1.3 L1 and L2 Regularization 3

2 Best Normalization Configuration **3**
2.1 Training and Validation 4
2.2 Evaluation 7

Overview

In this phase of the project, the performance of the final model from the previous phase was improved through the use of normalization techniques including dropout, batch normalization, and L1/L2 regularization.

1 Effects of Normalization

When used individually, each normalization technique either matched or slightly hindered the performance of the model. However, by testing each form of normalization and combining the best performing results in the model structure seen in Figure 1, a model stronger than that of the previous phase was attained.

1.1 Dropout

Unlike the other forms of normalization tested in this phase, adding a dropout layer to the model greatly increased the amount of epochs needed for training to converge, and most of the models tested for tuning dropout ran between 250-300 epochs. The performance results of various amounts of dropout are shown in Figure 2.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 150, 150, 3)]	0
rescaling (Rescaling)	(None, 150, 150, 3)	0
conv2d (Conv2D)	(None, 148, 148, 8)	224
max_pooling2d (MaxPooling2D)	(None, 37, 37, 8)	0
conv2d_1 (Conv2D)	(None, 35, 35, 7)	511
max_pooling2d_1 (MaxPooling2D)	(None, 8, 8, 7)	0
batch_normalization (BatchNormalization)	(None, 8, 8, 7)	28
conv2d_2 (Conv2D)	(None, 6, 6, 7)	448
max_pooling2d_2 (MaxPooling2D)	(None, 1, 1, 7)	0
flatten (Flatten)	(None, 7)	0
dropout (Dropout)	(None, 7)	0
dense (Dense)	(None, 1)	8

Total params: 1,219
Trainable params: 1,205
Non-trainable params: 14

Figure 1: The most effective structure in which normalization techniques were implemented.

Dropout

Amount	Validation Accuracy	Validation Loss	Validation Precision
0.5	0.85	0.3819	0.85
0.4	0.825	0.4403	0.8298
0.6	0.84	0.4321	0.8506
0.55	0.8550	0.3937	0.8554
0.56	0.88	0.3486	0.8587
0.57	0.88	0.363	0.8804

Figure 2: Validation results for various dropout configurations.

Batch Normalization

Placement	Validation Accuracy	Validation Loss	Validation Precision
After 3rd MaxPooling2D	0.78	0.4185	0.902
After 2nd MaxPooling2D	0.835	0.3651	0.8776
After 1st MaxPooling2D	0.855	0.3627	1

Figure 3: Validation results for various batch normalization configurations.

1.2 Batch Normalization

The introduction of batch normalization to the model significantly decreased the amount of time that the model needed to train and resulted in erratic validation metric curves. However, the performance of the model remained relatively strong, and validation precision results were extremely impressive, as seen in Figure 3.

1.3 L1 and L2 Regularization

In order for either form of regularization to allow the model to learn, both the dropout and batch normalization layers needed to be included in the model. Otherwise, the loss of the model would hover around 0.693 and the accuracy would remain at 0.5. Figure 4 thus shows the results of different L1 and L2 regularization amounts when all normalization techniques were implemented simultaneously. Using both forms of regularization proved to be most effective approach during experimentation.

2 Best Normalization Configuration

With validation loss as a guiding metric, as well as evaluation data acquired from each experiment, a model with the strongest validation and evaluation performance was selected

L1 and L2 Regularization

Amount	Validation Accuracy	Validation Loss	Validation Precision
L1=0.0005, L2=0.0005	0.85	0.431	0.8824
L1=0.0011, L2=0.0011	0.86	0.4108	0.8431
L1=0.0015, L2=0.0015	0.885	0.3695	0.8348
L1=0.0016, L2=0.0016	0.875	0.442	0.9474
L1=0.0014, L2=0.0014	0.845	0.4437	0.8367

Figure 4: Validation results for various L1 and L2 regularization configurations.

```
inputs = keras.Input(shape=(150, 150, 3))
x = data_augmentation(inputs)
x = layers.Rescaling(1./255)(inputs)
x = layers.Conv2D(filters=8, kernel_size=3)(x)
x = layers.MaxPooling2D(pool_size=4)(x)
x = layers.Conv2D(filters=7, kernel_size=3)(x)
x = layers.MaxPooling2D(pool_size=4)(x)
x = layers.BatchNormalization()(x)
x = layers.Conv2D(filters=7, kernel_size=3, kernel_regularizer=regularizers.l1_l
2(l1=0.0015, l2=0.0015))(x)
x = layers.MaxPooling2D(pool_size=4)(x)
x = layers.Flatten()(x)
x = layers.Dropout(0.56)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs=inputs, outputs=outputs)
model.summary()
```

Figure 5: The most effective normalization configuration found after experimentation.

(Figure 5). This model is identical to the experimental model with both L1 and L2 regularization amounts set to 0.0015.

2.1 Training and Validation

As in previous models, the training curves reflect a gradual movement towards overfitting the data. At the end of training, which occurred 30 epochs after the best validation loss was attained, the model had achieved a training accuracy of 0.86, loss of 0.356, and precision of 0.8454 (Figure 6).

The highest validation metrics achieved during training were an accuracy of 0.885 (notably higher than that of training), a loss of 0.3695, and a precision of 0.8348 (Figure 7).

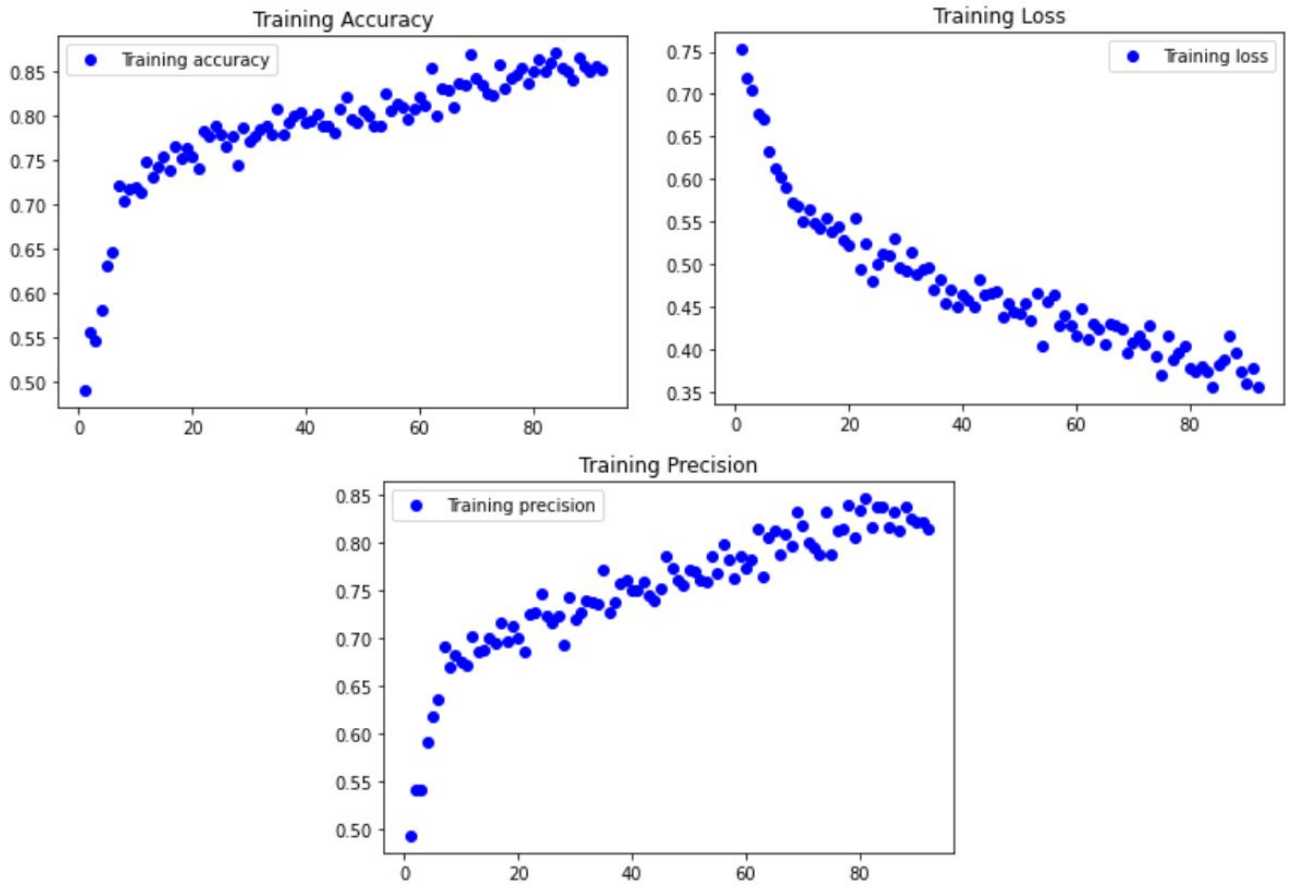

Figure 6: Training accuracy, loss, and precision of the best-performing model.

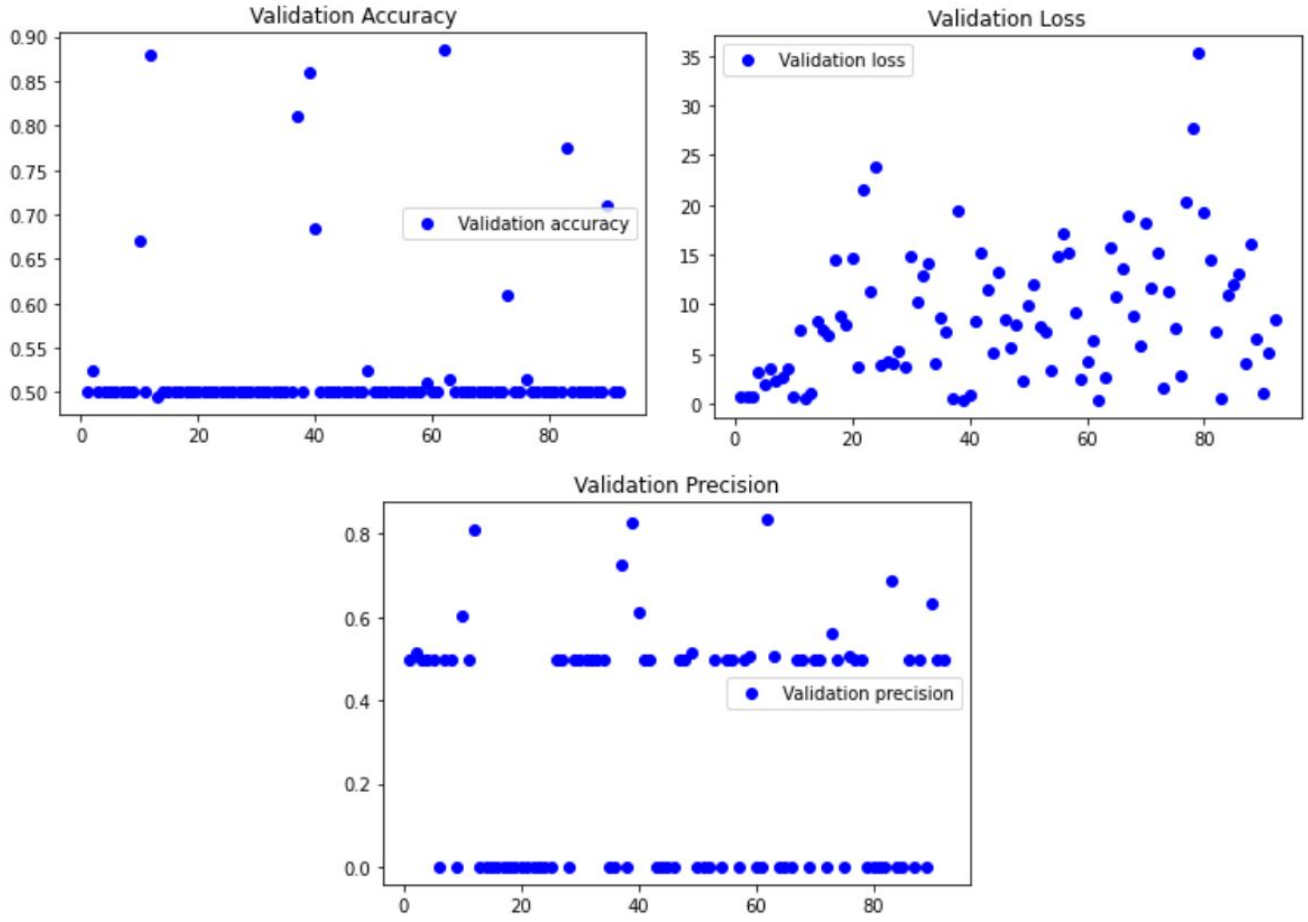

Figure 7: Validation accuracy, loss, and precision of the best-performing model.

```
loss: 0.3950 - accuracy: 0.8550 - precision: 0.8087
```

Figure 8: Results of testing the best-performing model.

2.2 Evaluation

Compared to the previous phase's model, the normalized model showed a remarkable improvement in all evaluation metrics. As seen in Figure 6, the evaluation accuracy reached 0.855, loss was reduced to 0.395, and precision increased to 0.8087. Therefore, each metric was improved by at least one point, while the loss dropped by four, during this phase.