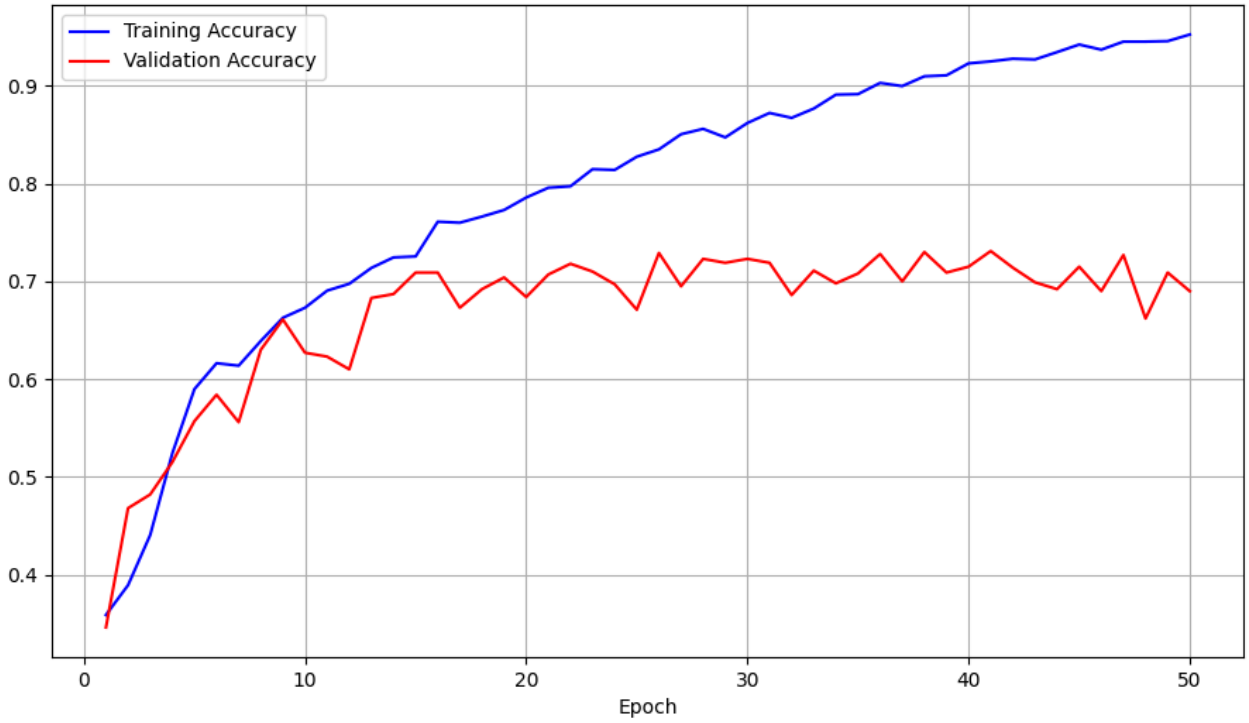
Initial Network

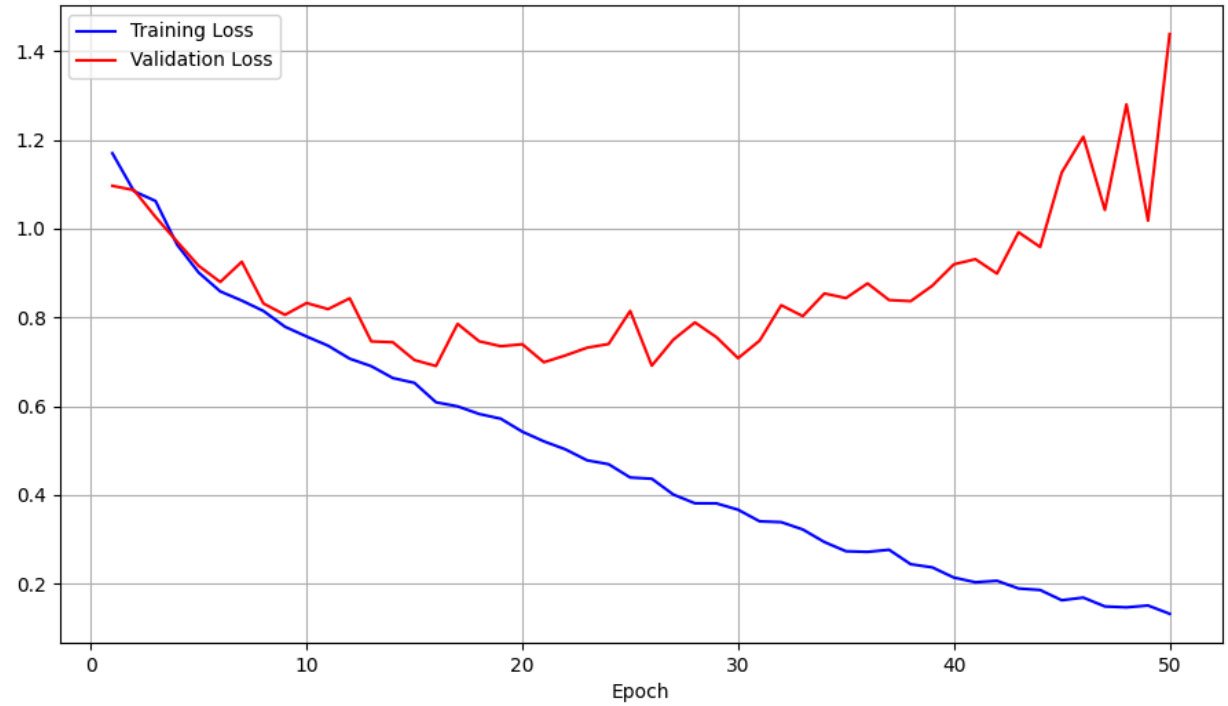
Layer (type)	Output Shape	Param #
rescaling (Rescaling)	(None, 150, 150, 3)	0
conv2d (Conv2D)	(None, 148, 148, 16)	448
max_pooling2d (MaxPooling2D)	(None, 74, 74, 16)	0
dropout (Dropout)	(None, 74, 74, 16)	0
conv2d_1 (Conv2D)	(None, 72, 72, 32)	4640
max_pooling2d_1 (MaxPooling2D)	(None, 36, 36, 32)	0
dropout_1 (Dropout)	(None, 36, 36, 32)	0
conv2d_2 (Conv2D)	(None, 34, 34, 64)	18496
max_pooling2d_2 (MaxPooling2D)	(None, 17, 17, 64)	0
dropout_2 (Dropout)	(None, 17, 17, 64)	0
conv2d_3 (Conv2D)	(None, 15, 15, 128)	73856
max_pooling2d_3 (MaxPooling2D)	(None, 7, 7, 128)	0
flatten (Flatten)	(None, 6272)	0
dense (Dense)	(None, 3)	18819

Total params: 116,259
Trainable params: 116,259
Non-trainable params: 0

Basic Model Evaluation:
loss: 1.1410 - accuracy: 0.7335

10 epochs:





Hyperparameter optimization strategy

We first added dropout layers, which prevents overfitting by setting the input of random neurons to zero. We ended up adding 3 dropout layers, and found that using dropout twice early on, then once at the end gave us the best results. The dropout percentage was also increased by 10% every time, going from 10% to 20% and ending at 30%. If the dropout percentages were increased by too much then the overall accuracy would end up going down over time. It is better to use a larger percentage dropout percentage with a larger number of neurons being created, however too high of a percentage caused the results to become worsened.