Image Classification
Cats vs Dogs

Image Classification using CNNs and Transfer Learning

Micha Birklbauer, 2020

Cats vs Dogs – The Dataset

- 12500 images of cats
- 12500 images of dogs
- 1000 images of each class used for testing
- 32 pixels minimum size
- 360 x 404 pixels average resolution

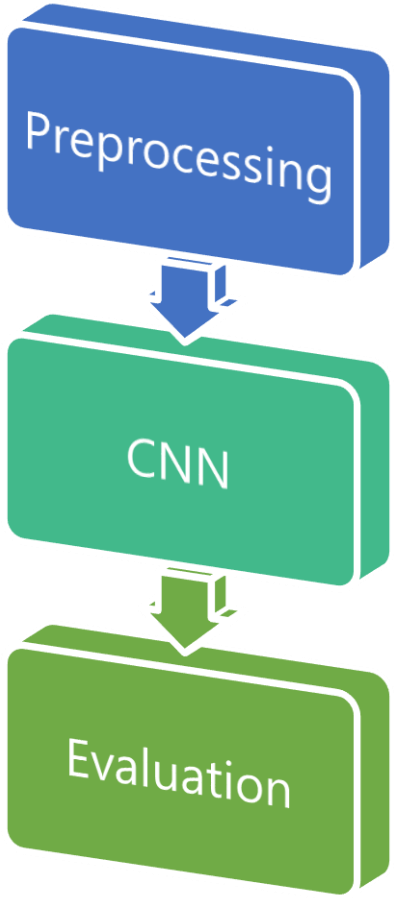
- Example Images




Pipeline

- Preprocessing images:
  - Generating RGB/Greyscale features
  - Mean-Substraction
  - Scaling
  - Data Augmentation
- Training with different model architectures:
  - Standard CNN
  - CNN with Batch-Normalization and Dropout
  - CNN with Batch-Normalization, Dropout and Regularization
  - Transfer Learning with ResNet
  - Transfer Learning with VGG16
- Evaluation on test images:
  - 1000 pictures with cats
  - 1000 pictures with dogs
- Models trained in total: 26

Workflow



CNN with Regularization trained with Greyscale Features – 0.763 Accuracy

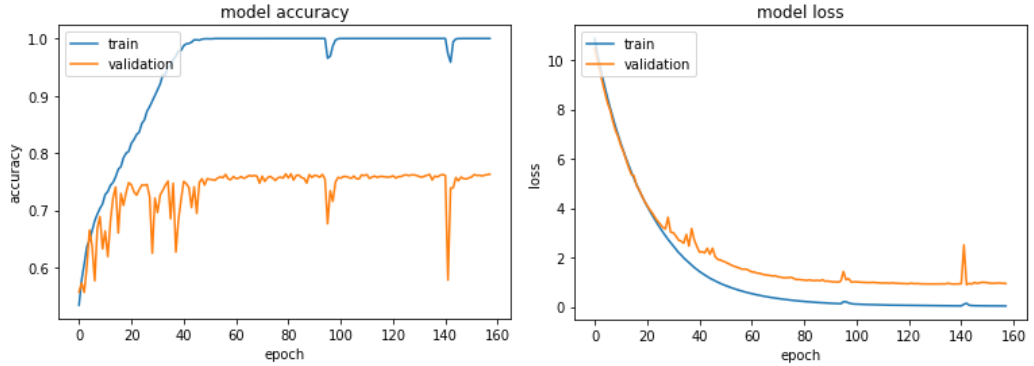
Model Summary

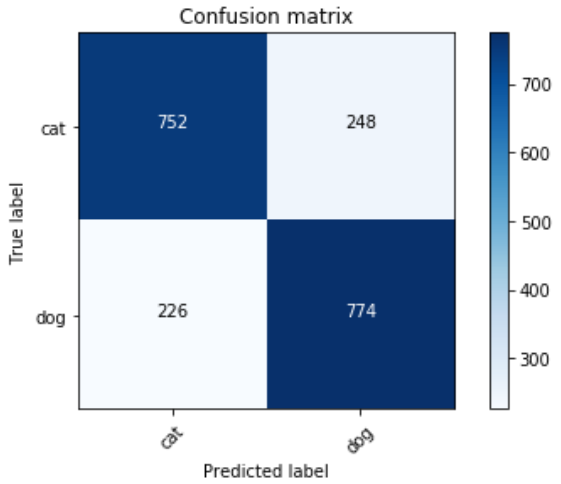
ValidationSplit: 0.3
Batchsize: 128
Epochs: 1000
Results:
Trainaccuracy: 100.0
Validationaccuracy: 76.28985643386841
Best Validation: 76.36231780052185

Modelarchitecture:

Layer (type)	Output Shape	Param #
conv2d_25 (Conv2D)	(None, 32, 32, 32)	832
conv2d_26 (Conv2D)	(None, 32, 32, 32)	25632
batch_normalization_8 (Batch	(None, 32, 32, 32)	128
max_pooling2d_20 (MaxPooling	(None, 16, 16, 32)	0
conv2d_27 (Conv2D)	(None, 16, 16, 64)	18496
max_pooling2d_21 (MaxPooling	(None, 8, 8, 64)	0
conv2d_28 (Conv2D)	(None, 8, 8, 128)	73856
max_pooling2d_22 (MaxPooling	(None, 4, 4, 128)	0
conv2d_29 (Conv2D)	(None, 4, 4, 128)	147584
max_pooling2d_23 (MaxPooling	(None, 2, 2, 128)	0
flatten_8 (Flatten)	(None, 512)	0
batch_normalization_9 (Batch	(None, 512)	2048
dense_24 (Dense)	(None, 512)	262656
dropout_12 (Dropout)	(None, 512)	0
dense_25 (Dense)	(None, 512)	262656
dropout_13 (Dropout)	(None, 512)	0
dense_26 (Dense)	(None, 2)	1026
activation_5 (Activation)	(None, 2)	0

Results





Accuracy on Test: 0.763

VGG16 with Fine Tuning and Mean-Substraction trained with RGB Features – 0.793 Accuracy

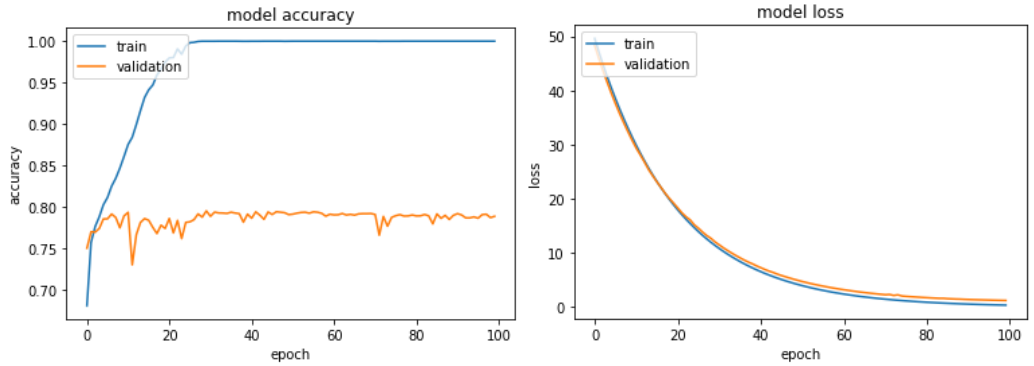
Model Summary

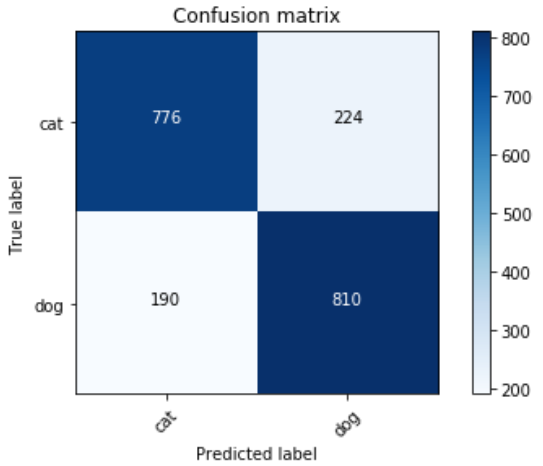
ValidationSplit: 0.3
Batchsize: 128
Epochs: 100
Results:
Trainaccuracy: 100.0
Validationaccuracy: 78.88405919075012
Best Validation: 79.53622937202454

Modelarchitecture:

Layer (type)	Output Shape	Param #
input_3 (InputLayer)	[(None, 32, 32, 3)]	0
15 pre-trained VGG16 layers		
block5_conv2 (Conv2D)	(None, 2, 2, 512)	2359808
block5_conv3 (Conv2D)	(None, 2, 2, 512)	2359808
block5_pool (MaxPooling2D)	(None, 1, 1, 512)	0
flatten_5 (Flatten)	(None, 512)	0
batch_normalization_5 (Batch	(None, 512)	2048
dense_15 (Dense)	(None, 4096)	2101248
dropout_8 (Dropout)	(None, 4096)	0
dense_16 (Dense)	(None, 4096)	16781312
dropout_9 (Dropout)	(None, 4096)	0
dense_17 (Dense)	(None, 2)	8194

Results





Accuracy on Test: 0.793

CNN with Batch-Normalization, Dropout and Data Augmentation trained with RGB Features – 0.7955 Accuracy

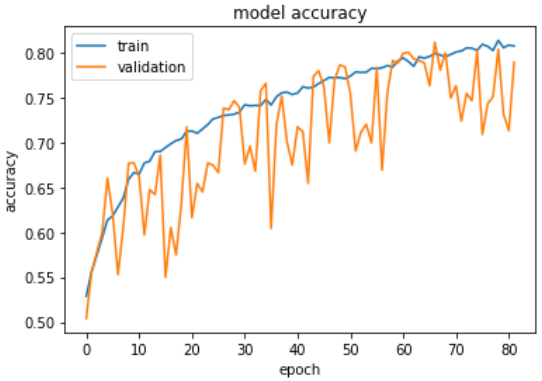
Model Summary

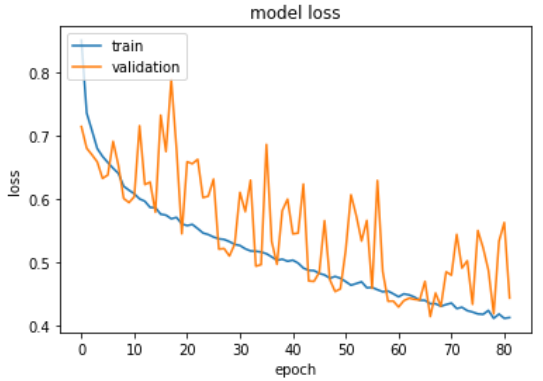
ValidationSplit: 0.3
Batchsize: 128
Epochs: 1000
Results:
Trainaccuracy: 80.71625232696533
Validationaccuracy: 78.89854907989502
Best Validation: 81.13043308258057

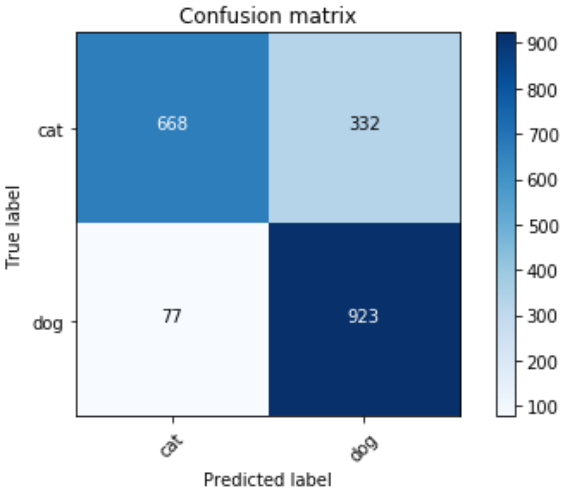
Modelarchitecture:

Layer (type)	Output Shape	Param #
conv2d_25 (Conv2D)	(None, 32, 32, 32)	2432
conv2d_26 (Conv2D)	(None, 32, 32, 32)	25632
batch_normalization_5 (Batch	(None, 32, 32, 32)	128
max_pooling2d_20 (MaxPooling	(None, 16, 16, 32)	0
conv2d_27 (Conv2D)	(None, 16, 16, 64)	18496
max_pooling2d_21 (MaxPooling	(None, 8, 8, 64)	0
conv2d_28 (Conv2D)	(None, 8, 8, 128)	73856
max_pooling2d_22 (MaxPooling	(None, 4, 4, 128)	0
conv2d_29 (Conv2D)	(None, 4, 4, 128)	147584
max_pooling2d_23 (MaxPooling	(None, 2, 2, 128)	0
flatten_7 (Flatten)	(None, 512)	0
batch_normalization_6 (Batch	(None, 512)	2048
dense_21 (Dense)	(None, 512)	262656
dropout_8 (Dropout)	(None, 512)	0
dense_22 (Dense)	(None, 512)	262656
dropout_9 (Dropout)	(None, 512)	0
dense_23 (Dense)	(None, 2)	1026
activation_5 (Activation)	(None, 2)	0

Results







Accuracy on Test: 0.7955

CNN with Regularization and Data Augmentation trained with RGB Features – 0.825 Accuracy

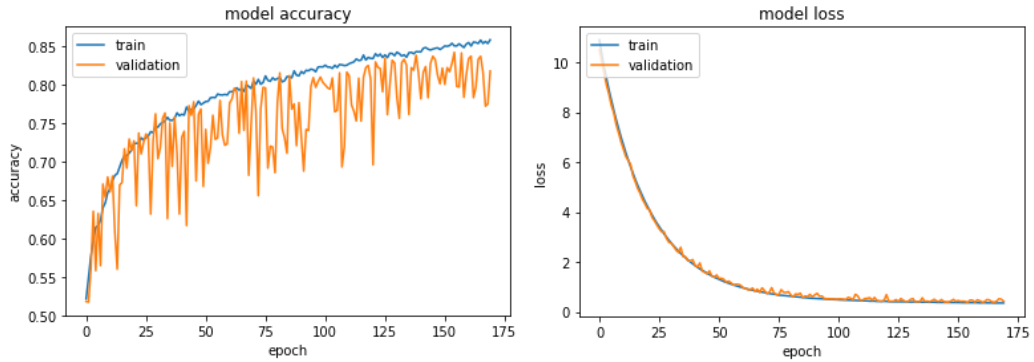
Model Summary

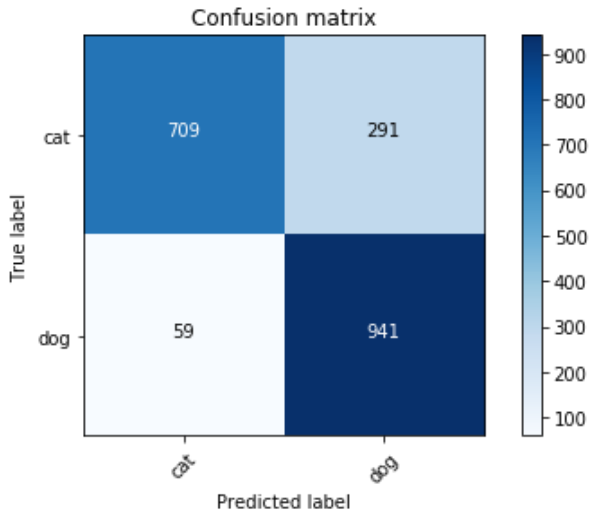
ValidationSplit: 0.3
Batchsize: 128
Epochs: 1000
Results:
Trainaccuracy: 85.88780164718628
Validationaccuracy: 81.81159496307373
Best Validation: 84.26086902618408

Modelarchitecture:

Layer (type)	Output Shape	Param #
conv2d_35 (Conv2D)	(None, 32, 32, 32)	2432
conv2d_36 (Conv2D)	(None, 32, 32, 32)	25632
batch_normalization_9 (Batch	(None, 32, 32, 32)	128
max_pooling2d_28 (MaxPooling	(None, 16, 16, 32)	0
conv2d_37 (Conv2D)	(None, 16, 16, 64)	18496
max_pooling2d_29 (MaxPooling	(None, 8, 8, 64)	0
conv2d_38 (Conv2D)	(None, 8, 8, 128)	73856
max_pooling2d_30 (MaxPooling	(None, 4, 4, 128)	0
conv2d_39 (Conv2D)	(None, 4, 4, 128)	147584
max_pooling2d_31 (MaxPooling	(None, 2, 2, 128)	0
flatten_9 (Flatten)	(None, 512)	0
batch_normalization_10 (Batc	(None, 512)	2048
dense_27 (Dense)	(None, 512)	262656
dropout_12 (Dropout)	(None, 512)	0
dense_28 (Dense)	(None, 512)	262656
dropout_13 (Dropout)	(None, 512)	0
dense_29 (Dense)	(None, 2)	1026
activation_7 (Activation)	(None, 2)	0

Results





Accuracy on Test: 0.825

VGG16 with Fine Tuning and Data Augmentation trained with Scaled RGB Features – 0.903 Accuracy

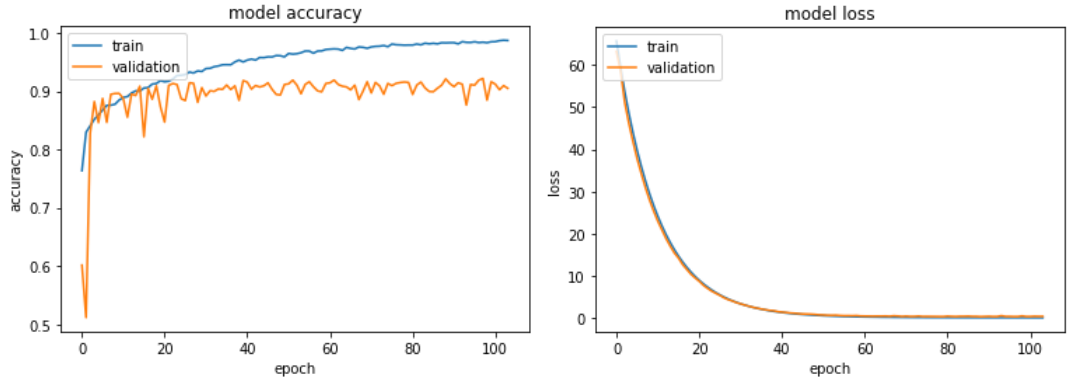
Model Summary

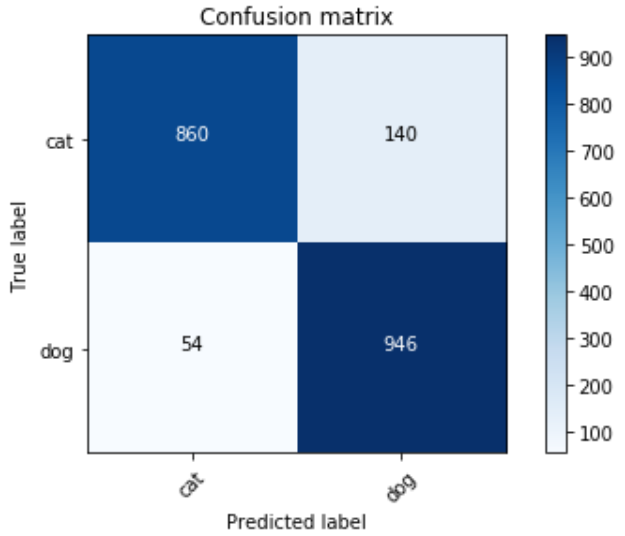
ValidationSplit: 0.3
Batchsize: 64
Epochs: 200
Results:
Trainaccuracy: 98.70915412902832
Validationaccuracy: 90.50724506378174
Best Validation: 92.18840599060059

Modelarchitecture:

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 64, 64, 3)]	0
15 pre-trained VGG15 layers		
block5_conv2 (Conv2D)	(None, 4, 4, 512)	2359808
block5_conv3 (Conv2D)	(None, 4, 4, 512)	2359808
block5_pool (MaxPooling2D)	(None, 2, 2, 512)	0
flatten (Flatten)	(None, 2048)	0
batch_normalization (BatchNo	(None, 2048)	8192
dense (Dense)	(None, 4096)	8392704
dropout (Dropout)	(None, 4096)	0
dense_1 (Dense)	(None, 4096)	16781312
dropout_1 (Dropout)	(None, 4096)	0
dense_2 (Dense)	(None, 2)	8194

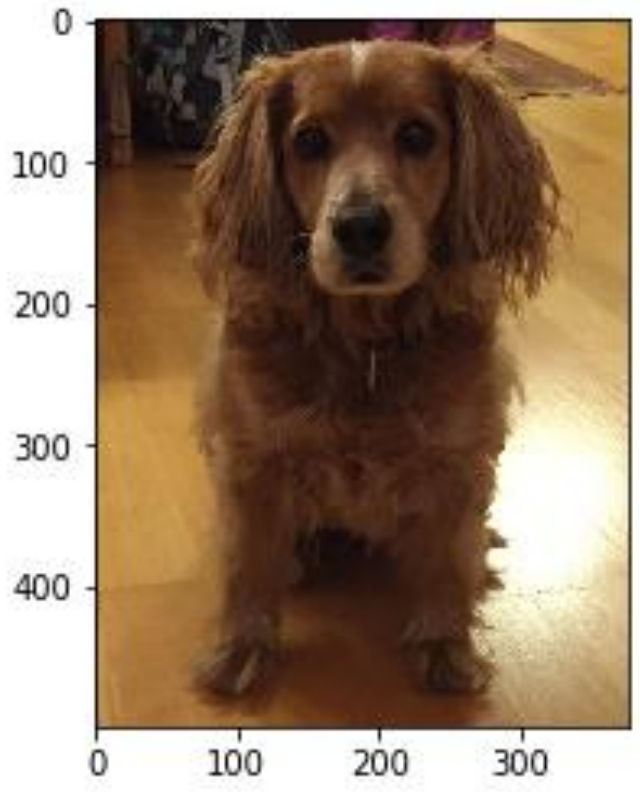
Results





Accuracy on Test: 0.903

Use Case I: Classifying my own Dogs

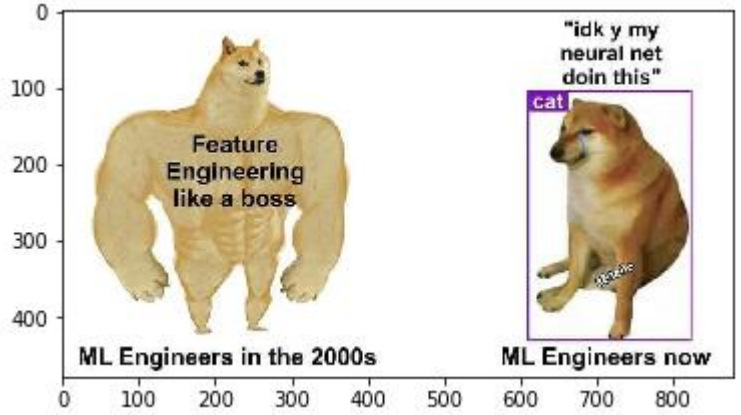


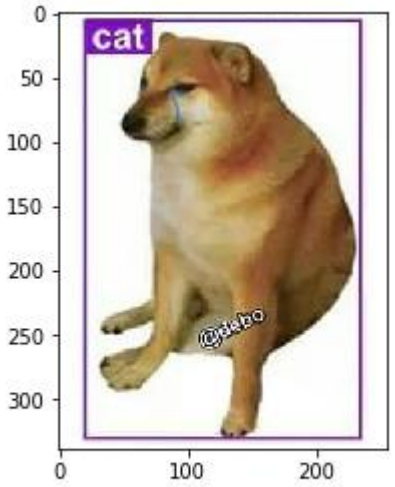
Predicted: dog

Predicted: dog

Predicted: dog

Use Case II: Predicting Cats and Dogs in Memes






```
Predicted:  dog
```


```
Predicted:  cat
```

Discussion & Conclusion

Discussion

- 4/5 of the best models were trained on scaled RGB features.
- 3/5 of the best models used transfer learning with VGG16 as base.
- For standard CNNs mean-substraction and greyscale features tended to work slightly better than normal RGB features.
- ResNet had an overall poor performance and in one case didn't learn at all.
- Adding batch-normalization and dropout always improved the results.
- Adding regularization always improved the results
- Adding data augmentation always improved the results.
- Scaling always improved the results (and maybe could have even further if it wasn't for technical constraints).

Conclusion

- Batch-Normalization, dropout, regularization, data augmentation and scaling all proved to be significant improvments in terms of accuracy for this classification task.
- Transfer learning works very well with VGG16, especially if one fine tunes the last layers as well.
- Mean-Substraction and greyscale features only improved results of simpler CNNs (without regularization and transfer learning).
- Transfer learning with ResNet did not work very well with this specific task.

More?

Read here!

Questions?

micha.birklbauer@gmail.com