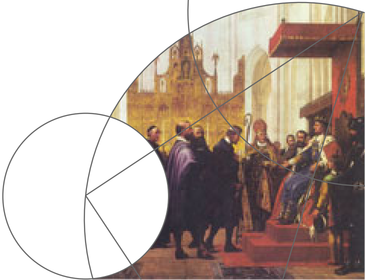
Faculty of Science

Classifying dog and cat images with Keras

Final project

Cecilie, Estrid, Maria, Louise
All group members have contributed evenly to this project

Niels Bohr Institute



Data

From Kaggle: 25,000 images of cats and dogs, half of each

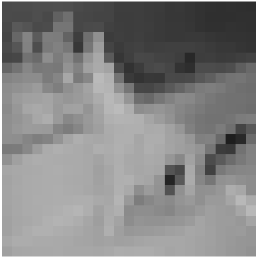
Goal: train an algorithm to classify whether images are cats or dogs

Preprocessing

Before

After, 100x100 pixels

After, 50x50 pixels



After, 25x25 pixels

Convolutional neural network

Models are built using Keras.

```
model = Sequential()
model.add(Conv2D(16, (3,3), input_shape=(IMG_SIZE, IMG_SIZE, 1), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(32, (3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.1))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(num_classes, activation='sigmoid'))
my_adam = keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False)
model.compile(loss='binary_crossentropy', optimizer=my_adam, metrics=['accuracy'])
```

Figure: Model 2 structure

```
Epoch 1/15
9000/9000 [==============================] - 27s 3ms/step - loss: 0.7162 - acc: 0.4990 - val_loss: 0.6851 - val_acc: 0.5190
Epoch 2/15
9000/9000 [==============================] - 24s 3ms/step - loss: 0.6818 - acc: 0.5597 - val_loss: 0.6742 - val_acc: 0.5410
Epoch 3/15
9000/9000 [==============================] - 24s 3ms/step - loss: 0.6646 - acc: 0.6176 - val_loss: 0.6526 - val_acc: 0.6090
Epoch 4/15
9000/9000 [==============================] - 24s 3ms/step - loss: 0.6416 - acc: 0.6650 - val_loss: 0.6324 - val_acc: 0.6500
```

Figure: Training a model

Keras models

Four different architectures:

Model 1	: C M F D D	Accuracy $(71.8 \pm 0.6)\%$
Model 2	: C M C M Dr F D D	Accuracy $(75.6 \pm 0.3)\%$
Model 2.1	: C C M Dr F D Dr D	Accuracy $(70.7 \pm 0.4)\%$
Model 3	: C M Dr C M Dr C M Dr F D D	Accuracy $(79.3 \pm 0.7)\%$

Trained on 10,000 images using 15 epochs

C = convolutional, M = max pooling, F = flatten,
D = dense, Dr = dropout

Hyperparameter optimization

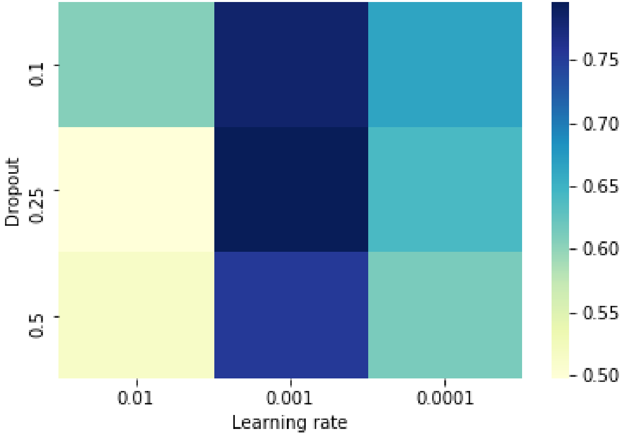


Figure: Validation accuracy as a function of dropout and learning rate.

Hyperparameter optimization

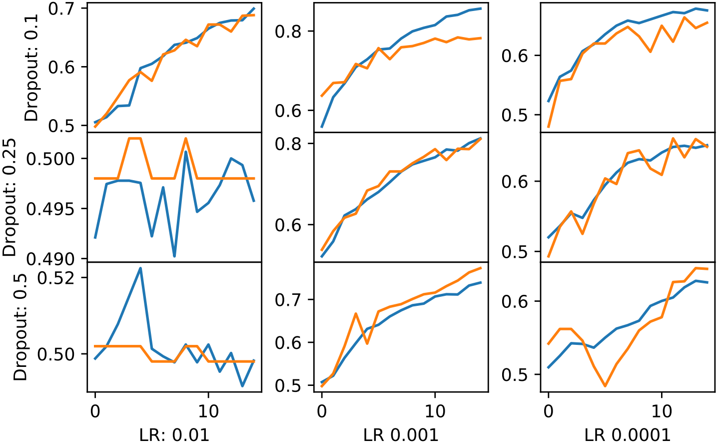


Figure: Accuracy as a function of epoch.
Blue = training accuracy, orange = validation accuracy

Finding the best optimizer

Optimizer	Optimal Epochs	Validation Accuracy
Adam	20	$81.2 \pm 0.6\%$
Adadelta	25	$75.7 \pm 0.9\%$
SGD	400+	$\sim 73\%$
AdaGrad	25	$79 \pm 1\%$

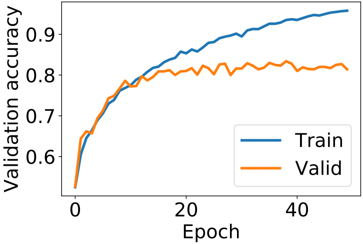


Figure: 10,000 pictures. Optimizer: Adam. LR: 0.001. Dropout: 0.25.

Results

Accuracy: $(85.3 \pm 0.5)\%$ after 30 epochs training on all 25,000 images.

Predict \ Truth	Cat	Dog	Total
Cat	1023	135	1158
Dog	201	1141	1342
Total	1224	1276	2500

Pretrained models can achieve an accuracy of up to 97%

Evaluation of the model

Examples of the images that the model guesses correctly with highest probability.

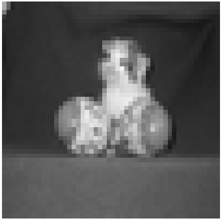
Prob: 99.99%

Prob: 99.99%

Prob: 99.99%

Evaluation of the model

Examples of images which the model guesses wrong.



Prob: 98% dog

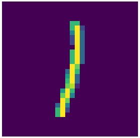
Prob: 97% cat

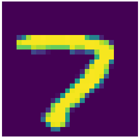
Prob: 66% dog

Prob: 97% dog

Testing the model on MNIST digit data

Test our best model on ones and sevens from the MNIST database.





Accuracy for cats & dogs: $(85.3 \pm 0.5)\%$
Accuracy for digits: $(99.84 \pm 0.03)\%$

References

Data: Kaggle
Model 1: Here
Model 2: Here (with modifications)
Model 2.1: Chollet's Architecture
Model 3: Malireddi's Architecture
General knowledge:
CNN and Image Classification
Developing a CNN
Hyperparameter Optimization with Keras
Guide to Hyperparameter Search
97 Percent Accuracy
MNIST digit: Dataset

Appendix

```
model = Sequential()
model.add(Conv2D(32, (3,3), input_shape=(IMG_SIZE, IMG_SIZE, 1), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))
my_adam = keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False)
model.compile(loss='binary_crossentropy', optimizer=my_adam, metrics=['accuracy'])
```

Figure: Model 1 structure

```
numOfFilters_1 = 16
numOfFilters_2 = 32
filterSize = 3
dropout = 0.1
numberOfDense = 128
lr = 0.001

model = Sequential()
model.add(Conv2D(numOfFilters_1, (filterSize,filterSize), input_shape=(IMG_SIZE, IMG_SIZE, 1), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(numOfFilters_2, (filterSize,filterSize), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(dropout))
model.add(Flatten())
model.add(Dense(numberOfDense, activation='relu'))
model.add(Dense(num_classes, activation='sigmoid'))
my_adam = keras.optimizers.Adam(lr=lr, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False)
model.compile(loss='binary_crossentropy', optimizer=my_adam, metrics=['accuracy'])
```

Figure: Model 2 structure

Appendix

```
dropout = 0.11127488091320181
filtersize = 3
lr= 0.0022693120181
numOfDense = 112
numOfFilters= 37
filterSize = 3

model_c = Sequential()
model_c.add(Conv2D(numOfFilters, (filterSize, filterSize), input_shape=(IMG_SIZE, IMG_SIZE, 1), activation='relu'))
model_c.add(Conv2D(numOfFilters, (filterSize, filterSize), activation='relu'))
model_c.add(MaxPooling2D(pool_size=(2, 2)))
model_c.add(Dropout(dropout))
model_c.add(Flatten())
model_c.add(Dense(numOfDense, activation='relu'))
model_c.add(Dropout(dropout))
model_c.add(Dense(num_classes, activation='softmax'))
my_adam = keras.optimizers.Adam(lr=lr, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False)
model_c.compile(loss='binary_crossentropy', optimizer=my_adam, metrics=['accuracy'])
```

Figure: Model 2.1 structure

Appendix

```
model_m = Sequential()
model_m.add(Conv2D(32, (3, 3), input_shape=(IMG_SIZE, IMG_SIZE, 1), activation='relu'))
model_m.add(MaxPooling2D(pool_size=(2, 2)))
model_m.add(Dropout(0.25))

model_m.add(Conv2D(64, (3, 3), activation='relu'))
model_m.add(MaxPooling2D(pool_size=(2, 2)))
model_m.add(Dropout(0.25))

model_m.add(Conv2D(128, (3, 3), activation='relu'))
model_m.add(MaxPooling2D(pool_size=(2, 2)))
model_m.add(Dropout(0.25))

model_m.add(Flatten())
model_m.add(Dense(64, activation='relu'))
model_m.add(Dense(num_classes, activation='sigmoid'))
print(model_m.summary())

my_adam = keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False)
model_m.compile(loss='binary_crossentropy', optimizer=my_adam, metrics=['accuracy'])
```

Figure: Model 3 structure

Appendix

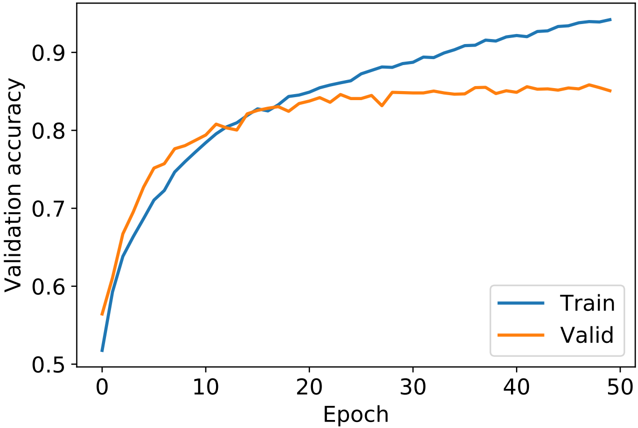


Figure: Model 3 with the Adam optimizer, learning rate 0.001, dropout rate 0.25 and run on all 25,000 pictures for 50 epochs.

Appendix

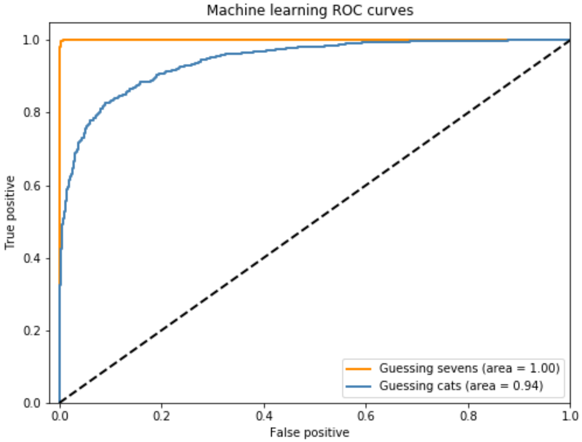


Figure: Roc curves for Model 3 using the cats/dogs dataset and the MNIST handwritten dataset.