# Mini-project Group 12

January 6, 2024

Oskar Lundqvist; osauli-0@student.ltu.se

Filip Renberg; filren-0@student.ltu.se

We are gonna do a simple lab where we train a model using Keras

We are gonna use the mnist dataset of digits imported from keras like we did in lab 1 and lab 4.

[ ]:

```
import logging
logging.getLogger('tensorflow').disabled = True # Disable warnings from␣
 ↪tensorflow
import numpy as np
import keras
from keras import layers
from keras.datasets import mnist
input_shape = (28,28,1)
num_classes = 10
```

We preprocess our data

[ ]:

```
(train_data, train_labels), (test_data, test_labels) = mnist.load_data()
#Normalize input values
train_data = train_data.astype("float32")/255
test_data = test_data.astype("float32")/255
#Make sure images have the correct shape(28,28,1)
train_data = np.expand_dims(train_data, -1)
test_data = np.expand_dims(test_data, -1)

#Converts class vectors to binary class matrices
train_labels = keras.utils.to_categorical(train_labels, num_classes)
test_labels = keras.utils.to_categorical(test_labels, num_classes)

print("training shape: ", train_data.shape)
print(train_data.shape[0], "number of training samples")
print(test_data.shape[0], "number of testing samples")
```

```
training shape:  (60000, 28, 28, 1)
60000 number of training samples
10000 number of testing samples
```

We create our convolutional neural network using Keras

[ ]:

```
def create_model(nr_conv2d = 1, nr_filter = 32, kernel_size = (3, 3),␣
↪model_summary = False):
    # Name the model based on the input values,
    # Sequential_nr_conv2d-nr_filter-kernel_size
    model_name=f"sequential_{nr_conv2d}-{nr_filter}-{kernel_size[0]}"

    hidden_layers = []

    hidden_layers.append(keras.Input(shape=input_shape)) # input layer

    for i in range(nr_conv2d):
        hidden_layers.append(layers.Conv2D(nr_filter, kernel_size=kernel_size,␣
↪activation="relu"))
        hidden_layers.append(layers.MaxPooling2D(pool_size=[2,2]))

        nr_filter*=2

    hidden_layers.append(layers.Flatten())
    hidden_layers.append(layers.Dropout(0.5)) #Prevents overfitting
    hidden_layers.append(layers.Dense(num_classes, activation="softmax"))

    model = keras.Sequential(hidden_layers, name=model_name)

    if model_summary == True:
        model.summary()
    else:
        print(f'Model: {model_name}')

    return model
```

Next we are training the model. We can use several different loss functions here, for this project we will use “categorical_crossentropy”, “poisson” and “binary_crossentropy”

[ ]:

```
n_epochs = 10
size_batch = 128

def model_training(pick, model):
    match pick:
        case 1:
            #Categorical loss function
            model.compile(loss="categorical_crossentropy", optimizer="adam",␣
↪metrics=['categorical_accuracy'])
            print("Loss Function: categorical_crossentropy")
            model.fit(train_data, train_labels, batch_size=size_batch,␣
↪epochs=n_epochs, validation_split=0.1, verbose=0)
        case 2:
            #Poisson loss function
            model.compile(loss="poisson", optimizer="adam",␣
↪metrics=['categorical_accuracy'])
            print("Loss Function: poisson")
            model.fit(train_data, train_labels, batch_size=size_batch,␣
↪epochs=n_epochs, validation_split=0.1, verbose=0)
        case 3:
            #Binary loss function
            model.compile(loss="binary_crossentropy", optimizer="adam",␣
↪metrics=['categorical_accuracy'])
            print("Loss Function: binary_crossentropy")
            model.fit(train_data, train_labels, batch_size=size_batch,␣
↪epochs=n_epochs, validation_split=0.1, verbose=0)
```

Define different values that we want to test

[ ]:

```
nr_conv2d_list = [1, 2, 3]
nr_filter_list = [8, 16, 32]
kernel_size_list = [1, 2, 3]

# We use this list to save the values for the best performing models
best_overall = [[0, 1]]
best_accuracy = [[0, 1]]
best_loss = [[0, 1]]
```

[ ]:

```
def best_performance(best_model):
    # Resulting best model:
    create_model(nr_conv2d=best_model[0][2], nr_filter=best_model[0][3],␣
↪model_summary=True)
    match best_model[0][4]:
        case 1:
            print("Using Categorical crossentropy loss function resulted in the␣
↪\nfollowing accuracy and loss:")
        case 2:
            print("Using Poisson loss function resulted in the \nfollowing␣
↪accuracy and loss:")
        case 3:
            print("Using binary crossentropy loss function resulted in the␣
↪\nfollowing accuracy and loss")
    print("|\taccuracy: \t", best_model[0][0], "\t|")
    print("|\tloss: \t\t", best_model[0][1], "\t|")
```

We mute model summary for the test prints but you can still see the model name, example:

`sequential_x1-x2-x3`

`x1` is the nr of conv2d layers, `x2` is the starting filter size, and `x3` is the kernel size.

[ ]:

```python
for i in nr_conv2d_list: # The nr conv2d layers
    for j in nr_filter_list: # The starting size of the conv2d filters
        for k in kernel_size_list: # Kernel size for the filters
            for n in range(3): # The loss function

                model = create_model(nr_conv2d=i, nr_filter=j, kernel_size=(k, k)) # You can also change the kernel size
                train = model_training(n+1, model)

                #Evaluate
                score = model.evaluate(test_data, test_labels, verbose=0)
                print("|\taccuracy: \t", score[1], "\t|")
                print("|\tloss: \t\t", score[0], "\t|")

                if (score[1] > best_overall[0][0]) & (score[0] < best_overall[0][1]): # best accuracy + loss
                    best_overall.pop()
                    best_overall.append((score[1], score[0], i, j, n+1))

                if (score[1] > best_accuracy[0][0]): # best accuracy
                    best_accuracy.pop()
                    best_accuracy.append((score[1], score[0], i, j, n+1))

                if (score[0] < best_loss[0][1]): # best loss
                    best_loss.pop()
                    best_loss.append((score[1], score[0], i, j, n+1))

                print("\n")

print("Finally done :)")
```

```
Model: sequential_1-8-1
Loss Function: categorical_crossentropy
|       accuracy:        0.9111999869346619     |
|       loss:            0.30452200770378113    |


Model: sequential_1-8-1
Loss Function: poisson
|       accuracy:        0.9142000079154968     |
|       loss:            0.13022562861442566    |


Model: sequential_1-8-1
Loss Function: binary_crossentropy
|       accuracy:        0.8944000005722046     |
|       loss:            0.08388058841228485    |
Model: sequential_1-8-2
Loss Function: categorical_crossentropy
|       accuracy:        0.9581999778747559     |
|       loss:            0.14477361738681793    |


Model: sequential_1-8-2
Loss Function: poisson
|       accuracy:        0.961899995803833      |
|       loss:            0.11375628411769867    |


Model: sequential_1-8-2
Loss Function: binary_crossentropy
|       accuracy:        0.954200029373169      |
|       loss:            0.039917927235364914   |


Model: sequential_1-8-3
Loss Function: categorical_crossentropy
|       accuracy:        0.9682999849319458     |
|       loss:            0.10456815361976624    |


Model: sequential_1-8-3
Loss Function: poisson
|       accuracy:        0.9692999720573425     |
|       loss:            0.11055201292037964    |


Model: sequential_1-8-3
Loss Function: binary_crossentropy
|       accuracy:        0.9617999792098999     |
|       loss:            0.031008990481495857   |


Model: sequential_1-16-1
Loss Function: categorical_crossentropy
|       accuracy:        0.9124000072479248     |
|       loss:            0.2945832312107086     |


Model: sequential_1-16-1
Loss Function: poisson
|       accuracy:        0.914900004863739      |
|       loss:            0.1297198385000229     |
Model: sequential_1-16-1
Loss Function: binary_crossentropy
|       accuracy:        0.9083999991416931     |
|       loss:            0.06987139582633972    |


Model: sequential_1-16-2
Loss Function: categorical_crossentropy
|       accuracy:        0.9692999720573425     |
|       loss:            0.10458052903413773    |


Model: sequential_1-16-2
Loss Function: poisson
|       accuracy:        0.9718999862670898     |
|       loss:            0.10959072411060333    |


Model: sequential_1-16-2
Loss Function: binary_crossentropy
|       accuracy:        0.9621000289916992     |
|       loss:            0.03141861408948898    |


Model: sequential_1-16-3
Loss Function: categorical_crossentropy
|       accuracy:        0.9761999845504761     |
|       loss:            0.07268141210079193    |


Model: sequential_1-16-3
Loss Function: poisson
|       accuracy:        0.9786999821662903     |
|       loss:            0.10692232847213745    |


Model: sequential_1-16-3
Loss Function: binary_crossentropy
|       accuracy:        0.9765999913215637     |
|       loss:            0.020543649792671204   |


Model: sequential_1-32-1
Loss Function: categorical_crossentropy
|       accuracy:        0.9176999926567078     |
|       loss:            0.28041988611221313    |
Model: sequential_1-32-1
Loss Function: poisson
|       accuracy:        0.9190000295639038     |
|       loss:            0.1282191425561905     |


Model: sequential_1-32-1
Loss Function: binary_crossentropy
|       accuracy:        0.9085000157356262     |
|       loss:            0.06618037074804306    |


Model: sequential_1-32-2
Loss Function: categorical_crossentropy
|       accuracy:        0.9760000109672546     |
|       loss:            0.07642963528633118    |


Model: sequential_1-32-2
Loss Function: poisson
|       accuracy:        0.9764000177383423     |
|       loss:            0.10754676163196564    |


Model: sequential_1-32-2
Loss Function: binary_crossentropy
|       accuracy:        0.9682000279426575     |
|       loss:            0.025774776935577393   |


Model: sequential_1-32-3
Loss Function: categorical_crossentropy
|       accuracy:        0.9824000000953674     |
|       loss:            0.056834980845451355   |


Model: sequential_1-32-3
Loss Function: poisson
|       accuracy:        0.9814000129699707     |
|       loss:            0.1056058332324028     |


Model: sequential_1-32-3
Loss Function: binary_crossentropy
|       accuracy:        0.9781000018119812     |
|       loss:            0.01797471195459366    |
Model: sequential_2-8-1
Loss Function: categorical_crossentropy
|       accuracy:        0.8546000123023987     |
|       loss:            0.4925113916397095     |


Model: sequential_2-8-1
Loss Function: poisson
|       accuracy:        0.8597000241279602     |
|       loss:            0.14700977504253387    |


Model: sequential_2-8-1
Loss Function: binary_crossentropy
|       accuracy:        0.8456000089645386     |
|       loss:            0.1057511568069458     |


Model: sequential_2-8-2
Loss Function: categorical_crossentropy
|       accuracy:        0.975600004196167      |
|       loss:            0.0781906396150589     |


Model: sequential_2-8-2
Loss Function: poisson
|       accuracy:        0.9761999845504761     |
|       loss:            0.10781963169574738    |


Model: sequential_2-8-2
Loss Function: binary_crossentropy
|       accuracy:        0.9746999740600586     |
|       loss:            0.023071149364113808   |


Model: sequential_2-8-3
Loss Function: categorical_crossentropy
|       accuracy:        0.9829999804496765     |
|       loss:            0.0523674450814724     |


Model: sequential_2-8-3
Loss Function: poisson
|       accuracy:        0.9837999939918518     |
|       loss:            0.1052933856844902     |
Model: sequential_2-8-3
Loss Function: binary_crossentropy
|       accuracy:        0.982699990272522      |
|       loss:            0.01610827073454857    |


Model: sequential_2-16-1
Loss Function: categorical_crossentropy
|       accuracy:        0.8632000088691711     |
|       loss:            0.45609068870544434    |


Model: sequential_2-16-1
Loss Function: poisson
|       accuracy:        0.8676999807357788     |
|       loss:            0.14288942515850067    |


Model: sequential_2-16-1
Loss Function: binary_crossentropy
|       accuracy:        0.8587999939918518     |
|       loss:            0.09436600655317307    |


Model: sequential_2-16-2
Loss Function: categorical_crossentropy
|       accuracy:        0.9837999939918518     |
|       loss:            0.048797789961099625   |


Model: sequential_2-16-2
Loss Function: poisson
|       accuracy:        0.9842000007629395     |
|       loss:            0.10497941821813583    |


Model: sequential_2-16-2
Loss Function: binary_crossentropy
|       accuracy:        0.9804999828338623     |
|       loss:            0.01698094978928566    |


Model: sequential_2-16-3
Loss Function: categorical_crossentropy
|       accuracy:        0.9890000224113464     |
|       loss:            0.03584475442767143    |
Model: sequential_2-16-3
Loss Function: poisson
|       accuracy:       0.9879000186920166      |
|       loss:           0.10372993350028992     |


Model: sequential_2-16-3
Loss Function: binary_crossentropy
|       accuracy:       0.987500011920929       |
|       loss:           0.011103938333690166    |


Model: sequential_2-32-1
Loss Function: categorical_crossentropy
|       accuracy:       0.8661999702453613      |
|       loss:           0.42086517810821533     |


Model: sequential_2-32-1
Loss Function: poisson
|       accuracy:       0.8622999787330627      |
|       loss:           0.1432160884141922      |


Model: sequential_2-32-1
Loss Function: binary_crossentropy
|       accuracy:       0.866100013256073       |
|       loss:           0.08706800639629364     |


Model: sequential_2-32-2
Loss Function: categorical_crossentropy
|       accuracy:       0.9861000180244446      |
|       loss:           0.04110102355480194     |


Model: sequential_2-32-2
Loss Function: poisson
|       accuracy:       0.9872000217437744      |
|       loss:           0.10391152650117874     |


Model: sequential_2-32-2
Loss Function: binary_crossentropy
|       accuracy:       0.98580002784729        |
|       loss:           0.011923989281058311    |
Model: sequential_2-32-3
Loss Function: categorical_crossentropy
|       accuracy:        0.9912999868392944     |
|       loss:            0.025549551472067833   |


Model: sequential_2-32-3
Loss Function: poisson
|       accuracy:        0.9907000064849854     |
|       loss:            0.10306503623723984    |


Model: sequential_2-32-3
Loss Function: binary_crossentropy
|       accuracy:        0.9904999732971191     |
|       loss:            0.008173882029950619   |


Model: sequential_3-8-1
Loss Function: categorical_crossentropy
|       accuracy:        0.566100001335144      |
|       loss:            1.3437367677688599     |


Model: sequential_3-8-1
Loss Function: poisson
|       accuracy:        0.6552000045776367     |
|       loss:            0.20856887102127075    |


Model: sequential_3-8-1
Loss Function: binary_crossentropy
|       accuracy:        0.633899986743927      |
|       loss:            0.19226402044296265    |


Model: sequential_3-8-2
Loss Function: categorical_crossentropy
|       accuracy:        0.9714000225067139     |
|       loss:            0.094182088971138      |


Model: sequential_3-8-2
Loss Function: poisson
|       accuracy:        0.9739999771118164     |
|       loss:            0.10907341539859772    |
Model: sequential_3-8-2
Loss Function: binary_crossentropy
|       accuracy:        0.9641000032424927      |
|       loss:            0.029729217290878296    |


Model: sequential_3-8-3
Loss Function: categorical_crossentropy
|       accuracy:        0.9692999720573425      |
|       loss:            0.09702785313129425     |


Model: sequential_3-8-3
Loss Function: poisson
|       accuracy:        0.968500018119812       |
|       loss:            0.11046236753463745     |


Model: sequential_3-8-3
Loss Function: binary_crossentropy
|       accuracy:        0.9638000130653381      |
|       loss:            0.03126277029514313     |


Model: sequential_3-16-1
Loss Function: categorical_crossentropy
|       accuracy:        0.6657000184059143      |
|       loss:            1.0308948755264282      |


Model: sequential_3-16-1
Loss Function: poisson
|       accuracy:        0.6869999766349792      |
|       loss:            0.19946211576461792     |


Model: sequential_3-16-1
Loss Function: binary_crossentropy
|       accuracy:        0.6517000198364258      |
|       loss:            0.18309813737869263     |


Model: sequential_3-16-2
Loss Function: categorical_crossentropy
|       accuracy:        0.9818000197410583      |
|       loss:            0.06009068712592125     |
Model: sequential_3-16-2
Loss Function: poisson
|       accuracy:        0.9811000227928162     |
|       loss:            0.10630549490451813    |


Model: sequential_3-16-2
Loss Function: binary_crossentropy
|       accuracy:        0.980400025844574      |
|       loss:            0.01608153246343136    |


Model: sequential_3-16-3
Loss Function: categorical_crossentropy
|       accuracy:        0.9818999767303467     |
|       loss:            0.061925407499074936   |


Model: sequential_3-16-3
Loss Function: poisson
|       accuracy:        0.9825999736785889     |
|       loss:            0.10590063780546188    |


Model: sequential_3-16-3
Loss Function: binary_crossentropy
|       accuracy:        0.9768000245094299     |
|       loss:            0.017312271520495415   |


Model: sequential_3-32-1
Loss Function: categorical_crossentropy
|       accuracy:        0.7143999934196472     |
|       loss:            0.9123744368553162     |


Model: sequential_3-32-1
Loss Function: poisson
|       accuracy:        0.6766999959945679     |
|       loss:            0.19979707896709442    |


Model: sequential_3-32-1
Loss Function: binary_crossentropy
|       accuracy:        0.6539999842643738     |
|       loss:            0.18011701107025146    |
Model: sequential_3-32-2
Loss Function: categorical_crossentropy
|       accuracy:        0.9887999892234802     |
|       loss:            0.037356436252593994   |


Model: sequential_3-32-2
Loss Function: poisson
|       accuracy:        0.9861999750137329     |
|       loss:            0.10472308099269867    |


Model: sequential_3-32-2
Loss Function: binary_crossentropy
|       accuracy:        0.9864000082015991     |
|       loss:            0.010712560266256332   |


Model: sequential_3-32-3
Loss Function: categorical_crossentropy
|       accuracy:        0.9894999861717224     |
|       loss:            0.037665847688913345   |


Model: sequential_3-32-3
Loss Function: poisson
|       accuracy:        0.9871000051498413     |
|       loss:            0.10423722118139267    |


Model: sequential_3-32-3
Loss Function: binary_crossentropy
|       accuracy:        0.9857000112533569     |
|       loss:            0.010064337402582169   |


Finally done :)
```

The best overall performing model:

```
[ ]: best_performance(best_overall)
```

```
Model: "sequential_2-32-3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_162 (Conv2D)         (None, 26, 26, 32)        320
 max_pooling2d_162 (MaxPool  (None, 13, 13, 32)        0
 ing2D)

 conv2d_163 (Conv2D)         (None, 11, 11, 64)        18496

 max_pooling2d_163 (MaxPool  (None, 5, 5, 64)          0
 ing2D)

 flatten_81 (Flatten)        (None, 1600)              0

 dropout_81 (Dropout)        (None, 1600)              0

 dense_81 (Dense)            (None, 10)                16010

=================================================================
Total params: 34826 (136.04 KB)
Trainable params: 34826 (136.04 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Using binary crossentropy loss function resulted in the
following accuracy and loss
|       accuracy:        0.9904999732971191     |
|       loss:            0.008173882029950619   |
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_162 (Conv2D)         (None, 26, 26, 32)        320

 max_pooling2d_162 (MaxPool  (None, 13, 13, 32)        0
 ing2D)

 conv2d_163 (Conv2D)         (None, 11, 11, 64)        18496

 max_pooling2d_163 (MaxPool  (None, 5, 5, 64)          0
 ing2D)

 flatten_81 (Flatten)        (None, 1600)              0

 dropout_81 (Dropout)        (None, 1600)              0

 dense_81 (Dense)            (None, 10)                16010

=================================================================
Total params: 34826 (136.04 KB)
Trainable params: 34826 (136.04 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Using binary crossentropy loss function resulted in the
```

```
following accuracy and loss
|       accuracy:        0.9904999732971191     |
|       loss:            0.008173882029950619   |
```

Most accurate model:

[ ]:

```
best_performance(best_accuracy)
```

```
Model: "sequential_2-32-3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_164 (Conv2D)         (None, 26, 26, 32)        320

 max_pooling2d_164 (MaxPool  (None, 13, 13, 32)        0
 ing2D)

 conv2d_165 (Conv2D)         (None, 11, 11, 64)        18496

 max_pooling2d_165 (MaxPool  (None, 5, 5, 64)          0
 ing2D)

 flatten_82 (Flatten)        (None, 1600)              0

 dropout_82 (Dropout)        (None, 1600)              0

 dense_82 (Dense)            (None, 10)                16010

=================================================================
Total params: 34826 (136.04 KB)
Trainable params: 34826 (136.04 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Using Categorical crossentropy loss function resulted in the
following accuracy and loss:
|       accuracy:        0.9912999868392944     |
|       loss:            0.025549551472067833   |
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_164 (Conv2D)         (None, 26, 26, 32)        320

 max_pooling2d_164 (MaxPool  (None, 13, 13, 32)        0
 ing2D)

 conv2d_165 (Conv2D)         (None, 11, 11, 64)        18496

 max_pooling2d_165 (MaxPool  (None, 5, 5, 64)          0
 ing2D)
 flatten_82 (Flatten)        (None, 1600)              0

 dropout_82 (Dropout)        (None, 1600)              0

 dense_82 (Dense)            (None, 10)                16010

=================================================================
Total params: 34826 (136.04 KB)
Trainable params: 34826 (136.04 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Using Categorical crossentropy loss function resulted in the
following accuracy and loss:
|       accuracy:        0.9912999868392944     |
|       loss:            0.025549551472067833   |
```

Best loss model:

[ ]:
```
best_performance(best_loss)
```

```
Model: "sequential_2-32-3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_166 (Conv2D)         (None, 26, 26, 32)        320

 max_pooling2d_166 (MaxPool  (None, 13, 13, 32)        0
 ing2D)

 conv2d_167 (Conv2D)         (None, 11, 11, 64)        18496

 max_pooling2d_167 (MaxPool  (None, 5, 5, 64)          0
 ing2D)

 flatten_83 (Flatten)        (None, 1600)              0

 dropout_83 (Dropout)        (None, 1600)              0

 dense_83 (Dense)            (None, 10)                16010

=================================================================
Total params: 34826 (136.04 KB)
Trainable params: 34826 (136.04 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Using binary crossentropy loss function resulted in the
following accuracy and loss
|       accuracy:        0.9904999732971191     |
|       loss:            0.008173882029950619   |
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_166 (Conv2D)         (None, 26, 26, 32)        320

 max_pooling2d_166 (MaxPool  (None, 13, 13, 32)        0
 ing2D)

 conv2d_167 (Conv2D)         (None, 11, 11, 64)        18496

 max_pooling2d_167 (MaxPool  (None, 5, 5, 64)          0
 ing2D)

 flatten_83 (Flatten)        (None, 1600)              0

 dropout_83 (Dropout)        (None, 1600)              0

 dense_83 (Dense)            (None, 10)                16010

=================================================================
Total params: 34826 (136.04 KB)
Trainable params: 34826 (136.04 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Using binary crossentropy loss function resulted in the
following accuracy and loss
|       accuracy:        0.9904999732971191     |
|       loss:            0.008173882029950619   |
```