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 model
best_model = [[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.

[ ]:
```
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_model[0][0]) & (score[0] <␣
 ↪best_model[0][1]):
                    best_model.pop()
                    best_model.append((score[1], score[0], i, j, n+1))

                print("\n")

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

```
Model: sequential_1-8-1
Loss Function: categorical_crossentropy
|       accuracy:        0.9128999710083008     |
|       loss:            0.2987935543060303     |


Model: sequential_1-8-1
Loss Function: poisson
|       accuracy:        0.9125999808311462     |
|       loss:            0.1306813806295395     |


Model: sequential_1-8-1
Loss Function: binary_crossentropy
|       accuracy:        0.902899980545044      |
|       loss:            0.07139772921800613    |


Model: sequential_1-8-2
Loss Function: categorical_crossentropy
|       accuracy:        0.9584000110626221     |
|       loss:            0.15162043273448944    |


Model: sequential_1-8-2
Loss Function: poisson
|       accuracy:        0.9440000057220459     |
|       loss:            0.11986614763736725    |
```

```
Model: sequential_1-8-2
Loss Function: binary_crossentropy
|       accuracy:        0.9598000049591064     |
|       loss:            0.03655649349093437    |


Model: sequential_1-8-3
Loss Function: categorical_crossentropy
|       accuracy:        0.9678000211715698     |
|       loss:            0.10980123281478882    |


Model: sequential_1-8-3
Loss Function: poisson
|       accuracy:        0.9702000021934509     |
|       loss:            0.11022289842367172    |


Model: sequential_1-8-3
Loss Function: binary_crossentropy
|       accuracy:        0.9629999995231628     |
|       loss:            0.03193367272615433    |


Model: sequential_1-16-1
Loss Function: categorical_crossentropy
|       accuracy:        0.9176999926567078     |
|       loss:            0.2894260883331299     |


Model: sequential_1-16-1
Loss Function: poisson
|       accuracy:        0.9151999950408936     |
|       loss:            0.12938936054706573    |


Model: sequential_1-16-1
Loss Function: binary_crossentropy
|       accuracy:        0.9031000137329102     |
|       loss:            0.06957709044218063    |


Model: sequential_1-16-2
Loss Function: categorical_crossentropy
|       accuracy:        0.9653000235557556     |
|       loss:            0.12096966803073883    |
```

```
Model: sequential_1-16-2
Loss Function: poisson
|       accuracy:        0.9696000218391418     |
|       loss:            0.11008027195930481    |


Model: sequential_1-16-2
Loss Function: binary_crossentropy
|       accuracy:        0.9588000178337097     |
|       loss:            0.03415116295218468    |


Model: sequential_1-16-3
Loss Function: categorical_crossentropy
|       accuracy:        0.979200005531311      |
|       loss:            0.06647966057062149    |


Model: sequential_1-16-3
Loss Function: poisson
|       accuracy:        0.9785000085830688     |
|       loss:            0.10694978386163712    |


Model: sequential_1-16-3
Loss Function: binary_crossentropy
|       accuracy:        0.9753999710083008     |
|       loss:            0.02192365936934948    |


Model: sequential_1-32-1
Loss Function: categorical_crossentropy
|       accuracy:        0.9179999828338623     |
|       loss:            0.2829427421092987     |


Model: sequential_1-32-1
Loss Function: poisson
|       accuracy:        0.9153000116348267     |
|       loss:            0.12937746942043304    |


Model: sequential_1-32-1
Loss Function: binary_crossentropy
|       accuracy:        0.9060999751091003     |
|       loss:            0.06932933628559113    |
```

```
Model: sequential_1-32-2
Loss Function: categorical_crossentropy
|       accuracy:        0.9764000177383423     |
|       loss:            0.07710807770490646    |


Model: sequential_1-32-2
Loss Function: poisson
|       accuracy:        0.9769999980926514     |
|       loss:            0.10761921107769012    |


Model: sequential_1-32-2
Loss Function: binary_crossentropy
|       accuracy:        0.9721999764442444     |
|       loss:            0.023090384900569916   |


Model: sequential_1-32-3
Loss Function: categorical_crossentropy
|       accuracy:        0.9825999736785889     |
|       loss:            0.0563504621386528     |


Model: sequential_1-32-3
Loss Function: poisson
|       accuracy:        0.982699990272522      |
|       loss:            0.10552532970905304    |


Model: sequential_1-32-3
Loss Function: binary_crossentropy
|       accuracy:        0.979200005531311      |
|       loss:            0.018165795132517815   |


Model: sequential_2-8-1
Loss Function: categorical_crossentropy
|       accuracy:        0.8482000231742859     |
|       loss:            0.5114039778709412     |


Model: sequential_2-8-1
Loss Function: poisson
|       accuracy:        0.8572999835014343     |
|       loss:            0.14986096322536469    |
```

```
Model: sequential_2-8-1
Loss Function: binary_crossentropy
|       accuracy:        0.8082000017166138     |
|       loss:            0.12720641493797302    |


Model: sequential_2-8-2
Loss Function: categorical_crossentropy
|       accuracy:        0.9775000214576721     |
|       loss:            0.07472839951515198    |


Model: sequential_2-8-2
Loss Function: poisson
|       accuracy:        0.9786999821662903     |
|       loss:            0.10688614845275879    |


Model: sequential_2-8-2
Loss Function: binary_crossentropy
|       accuracy:        0.975600004196167      |
|       loss:            0.022512884810566902   |


Model: sequential_2-8-3
Loss Function: categorical_crossentropy
|       accuracy:        0.9839000105857849     |
|       loss:            0.052849579602479935   |


Model: sequential_2-8-3
Loss Function: poisson
|       accuracy:        0.9821000099182129     |
|       loss:            0.1056961640715599     |


Model: sequential_2-8-3
Loss Function: binary_crossentropy
|       accuracy:        0.9814000129699707     |
|       loss:            0.016296260058879852   |


Model: sequential_2-16-1
Loss Function: categorical_crossentropy
|       accuracy:        0.8700000047683716     |
|       loss:            0.4206395149230957     |
```

```
Model: sequential_2-16-1
Loss Function: poisson
|       accuracy:        0.8639000058174133     |
|       loss:            0.14427681267261505    |


Model: sequential_2-16-1
Loss Function: binary_crossentropy
|       accuracy:        0.857200026512146      |
|       loss:            0.09680618345737457    |


Model: sequential_2-16-2
Loss Function: categorical_crossentropy
|       accuracy:        0.9843000173568726     |
|       loss:            0.05187729001045227    |


Model: sequential_2-16-2
Loss Function: poisson
|       accuracy:        0.9837999939918518     |
|       loss:            0.10517852753400803    |


Model: sequential_2-16-2
Loss Function: binary_crossentropy
|       accuracy:        0.9828000068664551     |
|       loss:            0.016292624175548553   |


Model: sequential_2-16-3
Loss Function: categorical_crossentropy
|       accuracy:        0.9884999990463257     |
|       loss:            0.036053456366062164   |


Model: sequential_2-16-3
Loss Function: poisson
|       accuracy:        0.9876000285148621     |
|       loss:            0.10367996990680695    |


Model: sequential_2-16-3
Loss Function: binary_crossentropy
|       accuracy:        0.9861000180244446     |
|       loss:            0.010561351664364338   |
```

```
Model: sequential_2-32-1
Loss Function: categorical_crossentropy
|       accuracy:        0.8700000047683716     |
|       loss:            0.4233608841896057     |


Model: sequential_2-32-1
Loss Function: poisson
|       accuracy:        0.8702999949455261     |
|       loss:            0.14102916419506073    |


Model: sequential_2-32-1
Loss Function: binary_crossentropy
|       accuracy:        0.885200023651123      |
|       loss:            0.07814329117536545    |


Model: sequential_2-32-2
Loss Function: categorical_crossentropy
|       accuracy:        0.9884999990463257     |
|       loss:            0.03674345090985298    |


Model: sequential_2-32-2
Loss Function: poisson
|       accuracy:        0.9865999817848206     |
|       loss:            0.10388823598623276    |


Model: sequential_2-32-2
Loss Function: binary_crossentropy
|       accuracy:        0.9864000082015991     |
|       loss:            0.012070694006979465   |


Model: sequential_2-32-3
Loss Function: categorical_crossentropy
|       accuracy:        0.9876000285148621     |
|       loss:            0.03534656763076782    |


Model: sequential_2-32-3
Loss Function: poisson
|       accuracy:        0.9911999702453613     |
|       loss:            0.10285915434360504    |
```

```
Model: sequential_2-32-3
Loss Function: binary_crossentropy
|       accuracy:        0.9901999831199646     |
|       loss:            0.007893931120634079   |


Model: sequential_3-8-1
Loss Function: categorical_crossentropy
|       accuracy:        0.6503999829292297     |
|       loss:            1.0868866443634033     |


Model: sequential_3-8-1
Loss Function: poisson
|       accuracy:        0.6401000022888184     |
|       loss:            0.21130110323429108    |


Model: sequential_3-8-1
Loss Function: binary_crossentropy
|       accuracy:        0.6452000141143799     |
|       loss:            0.19146348536014557    |


Model: sequential_3-8-2
Loss Function: categorical_crossentropy
|       accuracy:        0.9717000126838684     |
|       loss:            0.0974355936050415     |


Model: sequential_3-8-2
Loss Function: poisson
|       accuracy:        0.9739000201225281     |
|       loss:            0.10887933522462845    |


Model: sequential_3-8-2
Loss Function: binary_crossentropy
|       accuracy:        0.9664999842643738     |
|       loss:            0.02884790301322937    |


Model: sequential_3-8-3
Loss Function: categorical_crossentropy
|       accuracy:        0.9714999794960022     |
|       loss:            0.09493064880371094    |
```

```
Model: sequential_3-8-3
Loss Function: poisson
|       accuracy:        0.9696999788284302     |
|       loss:            0.11098623275756836    |


Model: sequential_3-8-3
Loss Function: binary_crossentropy
|       accuracy:        0.9646999835968018     |
|       loss:            0.030567891895771027   |


Model: sequential_3-16-1
Loss Function: categorical_crossentropy
|       accuracy:        0.6650999784469604     |
|       loss:            1.0373120307922363     |


Model: sequential_3-16-1
Loss Function: poisson
|       accuracy:        0.6574000120162964     |
|       loss:            0.20624008774757385    |


Model: sequential_3-16-1
Loss Function: binary_crossentropy
|       accuracy:        0.650600016117096      |
|       loss:            0.1845005750656128     |


Model: sequential_3-16-2
Loss Function: categorical_crossentropy
|       accuracy:        0.9825000166893005     |
|       loss:            0.05720340088009834    |


Model: sequential_3-16-2
Loss Function: poisson
|       accuracy:        0.9825999736785889     |
|       loss:            0.10575384646654129    |


Model: sequential_3-16-2
Loss Function: binary_crossentropy
|       accuracy:        0.9799000024795532     |
|       loss:            0.016392391175031662   |
```

```
Model: sequential_3-16-3
Loss Function: categorical_crossentropy
|       accuracy:        0.9807999730110168     |
|       loss:            0.06029291823506355    |


Model: sequential_3-16-3
Loss Function: poisson
|       accuracy:        0.9812999963760376     |
|       loss:            0.10625935345888138    |


Model: sequential_3-16-3
Loss Function: binary_crossentropy
|       accuracy:        0.9819999933242798     |
|       loss:            0.014470896683633327   |


Model: sequential_3-32-1
Loss Function: categorical_crossentropy
|       accuracy:        0.7027000188827515     |
|       loss:            0.9355225563049316     |


Model: sequential_3-32-1
Loss Function: poisson
|       accuracy:        0.6965000033378601     |
|       loss:            0.19633089005947113    |


Model: sequential_3-32-1
Loss Function: binary_crossentropy
|       accuracy:        0.666100025177002      |
|       loss:            0.1765071451663971     |


Model: sequential_3-32-2
Loss Function: categorical_crossentropy
|       accuracy:        0.9879000186920166     |
|       loss:            0.03969316557049751    |


Model: sequential_3-32-2
Loss Function: poisson
|       accuracy:        0.9889000058174133     |
|       loss:            0.10389383882284164    |
```

```
Model: sequential_3-32-2
Loss Function: binary_crossentropy
|       accuracy:        0.986299991607666      |
|       loss:            0.011301561258733273   |


Model: sequential_3-32-3
Loss Function: categorical_crossentropy
|       accuracy:        0.9872000217437744     |
|       loss:            0.04405169561505318    |


Model: sequential_3-32-3
Loss Function: poisson
|       accuracy:        0.9873999953269958     |
|       loss:            0.10412595421075821    |


Model: sequential_3-32-3
Loss Function: binary_crossentropy
|       accuracy:        0.9861999750137329     |
|       loss:            0.010200412943959236   |


Finally done :)
```

The following is the best performing model from our tests:

[ ]:
```
best_performance(best_model)
```

```
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:        0.9901999831199646     |
|       loss:            0.007893931120634079   |
_________________________________________________________________
 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:        0.9901999831199646     |
|       loss:            0.007893931120634079   |
```