Mini-project Group 12

January 2, 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.

[ ]:

```
from silence_tensorflow import silence_tensorflow
silence_tensorflow() # Remove non-important tensorflow warnings

import numpy as np
import keras
from keras import layers
from keras.datasets import mnist
input_shape = (28,28,1)
num_classes = 10
```

```
<IPython.core.display.HTML object>
```

```
WARNING:tensorflow:From C:\Users\oskar\AppData\Local\Packages\PythonSoftwareFoun
dation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\site-
packages\keras\src\losses.py:2976: The name
tf.losses.sparse_softmax_cross_entropy is deprecated. Please use
tf.compat.v1.losses.sparse_softmax_cross_entropy instead.
```

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)
```

```python
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

[ ]:

```python
def create_model(nr_conv2d = 1, nr_filter = 32, kernel_size = (3, 3), model_summary = False):
    # Name the model based on the input values,
    # Model_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”

[ ]:

```python
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|")
```

[ ]:

```python
print("We mute model summary for the test prints but you can still see the model name:")
print("first value is the nr of conv2d layers, second value is the starting filter size.")
print("The third value is the kernel size but we don't change it in our current tests.\n")

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 :)")
```

```
We mute model summary for the test prints but you can still see the model name:
first value is the nr of conv2d layers, second value is the starting filter
size.
The third value is the kernel size but we don't change it in our current tests.

Model: sequential_1-8-1
Loss Function: categorical_crossentropy
|       accuracy:        0.9045000076293945      |
|       loss:            0.34677305817604065     |


Model: sequential_1-8-1
Loss Function: poisson
|       accuracy:        0.9099000096321106      |
|       loss:            0.13107535243034363     |
Model: sequential_1-8-1
Loss Function: binary_crossentropy
|       accuracy:        0.9020000100135803     |
|       loss:            0.07336410880088806    |


Model: sequential_1-8-2
Loss Function: categorical_crossentropy
|       accuracy:        0.958899974822998      |
|       loss:            0.14431360363960266    |


Model: sequential_1-8-2
Loss Function: poisson
|       accuracy:        0.9645000100135803     |
|       loss:            0.11252903193235397    |


Model: sequential_1-8-2
Loss Function: binary_crossentropy
|       accuracy:        0.9466999769210815     |
|       loss:            0.04450652003288269    |


Model: sequential_1-8-3
Loss Function: categorical_crossentropy
|       accuracy:        0.9711999893188477     |
|       loss:            0.09822211414575577    |


Model: sequential_1-8-3
Loss Function: poisson
|       accuracy:        0.9735999703407288     |
|       loss:            0.10903801023960114    |


Model: sequential_1-8-3
Loss Function: binary_crossentropy
|       accuracy:        0.965499997138977      |
|       loss:            0.02963523007929325    |


Model: sequential_1-16-1
Loss Function: categorical_crossentropy
|       accuracy:        0.9160000085830688     |
|       loss:            0.2961479127407074     |
Model: sequential_1-16-1
Loss Function: poisson
|       accuracy:        0.9136000275611877     |
|       loss:            0.12960928678512573    |


Model: sequential_1-16-1
Loss Function: binary_crossentropy
|       accuracy:        0.9064000248908997     |
|       loss:            0.06932251155376434    |


Model: sequential_1-16-2
Loss Function: categorical_crossentropy
|       accuracy:        0.9725000262260437     |
|       loss:            0.09533493220806122    |


Model: sequential_1-16-2
Loss Function: poisson
|       accuracy:        0.9718999862670898     |
|       loss:            0.10945045202970505    |


Model: sequential_1-16-2
Loss Function: binary_crossentropy
|       accuracy:        0.96670001745224       |
|       loss:            0.028368938714265823   |


Model: sequential_1-16-3
Loss Function: categorical_crossentropy
|       accuracy:        0.9800000190734863     |
|       loss:            0.06785483658313751    |


Model: sequential_1-16-3
Loss Function: poisson
|       accuracy:        0.9778000116348267     |
|       loss:            0.10673431307077408    |


Model: sequential_1-16-3
Loss Function: binary_crossentropy
|       accuracy:        0.9771999716758728     |
|       loss:            0.02028021775186062    |
Model: sequential_1-32-1
Loss Function: categorical_crossentropy
|       accuracy:        0.9169999957084656     |
|       loss:            0.2856709957122803     |


Model: sequential_1-32-1
Loss Function: poisson
|       accuracy:        0.9178000092506409     |
|       loss:            0.1284821480512619     |


Model: sequential_1-32-1
Loss Function: binary_crossentropy
|       accuracy:        0.9103000164031982     |
|       loss:            0.06726555526256561    |


Model: sequential_1-32-2
Loss Function: categorical_crossentropy
|       accuracy:        0.9753000140190125     |
|       loss:            0.08092426508665085    |


Model: sequential_1-32-2
Loss Function: poisson
|       accuracy:        0.9750999808311462     |
|       loss:            0.10779019445180893    |


Model: sequential_1-32-2
Loss Function: binary_crossentropy
|       accuracy:        0.9740999937057495     |
|       loss:            0.02248363569378853    |


Model: sequential_1-32-3
Loss Function: categorical_crossentropy
|       accuracy:        0.9822999835014343     |
|       loss:            0.05565842241048813    |


Model: sequential_1-32-3
Loss Function: poisson
|       accuracy:        0.9824000000953674     |
|       loss:            0.10548123717308044    |
Model: sequential_1-32-3
Loss Function: binary_crossentropy
|       accuracy:        0.9793000221252441     |
|       loss:            0.01774715445935726    |


Model: sequential_2-8-1
Loss Function: categorical_crossentropy
|       accuracy:        0.862500011920929      |
|       loss:            0.4641565978527069     |


Model: sequential_2-8-1
Loss Function: poisson
|       accuracy:        0.8756999969482422     |
|       loss:            0.1407691091299057     |


Model: sequential_2-8-1
Loss Function: binary_crossentropy
|       accuracy:        0.8684999942779541     |
|       loss:            0.09200688451528549    |


Model: sequential_2-8-2
Loss Function: categorical_crossentropy
|       accuracy:        0.9779999852180481     |
|       loss:            0.06890183687210083    |


Model: sequential_2-8-2
Loss Function: poisson
|       accuracy:        0.9779000282287598     |
|       loss:            0.10701524466276169    |


Model: sequential_2-8-2
Loss Function: binary_crossentropy
|       accuracy:        0.9751999974250793     |
|       loss:            0.023338187485933304   |


Model: sequential_2-8-3
Loss Function: categorical_crossentropy
|       accuracy:        0.9829000234603882     |
|       loss:            0.055518683046102524   |
Model: sequential_2-8-3
Loss Function: poisson
|       accuracy:        0.984000027179718      |
|       loss:            0.1050797626376152     |


Model: sequential_2-8-3
Loss Function: binary_crossentropy
|       accuracy:        0.9811000227928162     |
|       loss:            0.016874121502041817   |


Model: sequential_2-16-1
Loss Function: categorical_crossentropy
|       accuracy:        0.8657000064849854     |
|       loss:            0.43340224027633667    |


Model: sequential_2-16-1
Loss Function: poisson
|       accuracy:        0.8867999911308289     |
|       loss:            0.1376599669456482     |


Model: sequential_2-16-1
Loss Function: binary_crossentropy
|       accuracy:        0.8590999841690063     |
|       loss:            0.09501325339078903    |


Model: sequential_2-16-2
Loss Function: categorical_crossentropy
|       accuracy:        0.9840999841690063     |
|       loss:            0.05054925009608269    |


Model: sequential_2-16-2
Loss Function: poisson
|       accuracy:        0.9824000000953674     |
|       loss:            0.10543941706418991    |


Model: sequential_2-16-2
Loss Function: binary_crossentropy
|       accuracy:        0.9825999736785889     |
|       loss:            0.015554818324744701   |
Model: sequential_2-16-3
Loss Function: categorical_crossentropy
|       accuracy:        0.9876000285148621     |
|       loss:            0.035443928092718124   |


Model: sequential_2-16-3
Loss Function: poisson
|       accuracy:        0.9896000027656555     |
|       loss:            0.10363186150789261    |


Model: sequential_2-16-3
Loss Function: binary_crossentropy
|       accuracy:        0.9871000051498413     |
|       loss:            0.010172258131206036   |


Model: sequential_2-32-1
Loss Function: categorical_crossentropy
|       accuracy:        0.8823000192642212     |
|       loss:            0.38418707251548767    |


Model: sequential_2-32-1
Loss Function: poisson
|       accuracy:        0.8726999759674072     |
|       loss:            0.14082711935043335    |


Model: sequential_2-32-1
Loss Function: binary_crossentropy
|       accuracy:        0.8611999750137329     |
|       loss:            0.09058661758899689    |


Model: sequential_2-32-2
Loss Function: categorical_crossentropy
|       accuracy:        0.9868000149726868     |
|       loss:            0.040284544229507446   |


Model: sequential_2-32-2
Loss Function: poisson
|       accuracy:        0.9854999780654907     |
|       loss:            0.1041753813624382     |
Model: sequential_2-32-2
Loss Function: binary_crossentropy
|       accuracy:        0.9861999750137329     |
|       loss:            0.01182855386286974    |


Model: sequential_2-32-3
Loss Function: categorical_crossentropy
|       accuracy:        0.9902999997138977     |
|       loss:            0.029759705066680908   |


Model: sequential_2-32-3
Loss Function: poisson
|       accuracy:        0.9919000267982483     |
|       loss:            0.10243698954582214    |


Model: sequential_2-32-3
Loss Function: binary_crossentropy
|       accuracy:        0.989799976348877      |
|       loss:            0.008357501588761806   |


Model: sequential_3-8-1
Loss Function: categorical_crossentropy
|       accuracy:        0.6349999904632568     |
|       loss:            1.1341913938522339     |


Model: sequential_3-8-1
Loss Function: poisson
|       accuracy:        0.6380000114440918     |
|       loss:            0.21298415958881378    |


Model: sequential_3-8-1
Loss Function: binary_crossentropy
|       accuracy:        0.6355000138282776     |
|       loss:            0.19037678837776184    |


Model: sequential_3-8-2
Loss Function: categorical_crossentropy
|       accuracy:        0.9696000218391418     |
|       loss:            0.10524994134902954    |
Model: sequential_3-8-2
Loss Function: poisson
|       accuracy:        0.9692999720573425     |
|       loss:            0.11103629320859909    |


Model: sequential_3-8-2
Loss Function: binary_crossentropy
|       accuracy:        0.9581999778747559     |
|       loss:            0.035895027220249176   |


Model: sequential_3-8-3
Loss Function: categorical_crossentropy
|       accuracy:        0.9672999978065491     |
|       loss:            0.11193626374006271    |


Model: sequential_3-8-3
Loss Function: poisson
|       accuracy:        0.9697999954223633     |
|       loss:            0.11056715250015259    |


Model: sequential_3-8-3
Loss Function: binary_crossentropy
|       accuracy:        0.9596999883651733     |
|       loss:            0.038717880845069885   |


Model: sequential_3-16-1
Loss Function: categorical_crossentropy
|       accuracy:        0.661899983882904      |
|       loss:            1.0449005365371704     |


Model: sequential_3-16-1
Loss Function: poisson
|       accuracy:        0.6635000109672546     |
|       loss:            0.20406877994537354    |


Model: sequential_3-16-1
Loss Function: binary_crossentropy
|       accuracy:        0.6579999923706055     |
|       loss:            0.18362551927566528    |
Model: sequential_3-16-2
Loss Function: categorical_crossentropy
|       accuracy:        0.9815999865531921     |
|       loss:            0.05796007812023163    |


Model: sequential_3-16-2
Loss Function: poisson
|       accuracy:        0.9830999970436096     |
|       loss:            0.1051589846611023     |


Model: sequential_3-16-2
Loss Function: binary_crossentropy
|       accuracy:        0.9789000153541565     |
|       loss:            0.01722176931798458    |


Model: sequential_3-16-3
Loss Function: categorical_crossentropy
|       accuracy:        0.9839000105857849     |
|       loss:            0.05719917640089989    |


Model: sequential_3-16-3
Loss Function: poisson
|       accuracy:        0.984000027179718      |
|       loss:            0.1055717021226883     |


Model: sequential_3-16-3
Loss Function: binary_crossentropy
|       accuracy:        0.9797999858856201     |
|       loss:            0.016070352867245674   |


Model: sequential_3-32-1
Loss Function: categorical_crossentropy
|       accuracy:        0.6797999739646912     |
|       loss:            0.9819020628929138     |


Model: sequential_3-32-1
Loss Function: poisson
|       accuracy:        0.6796000003814697     |
|       loss:            0.19981011748313904    |
Model: sequential_3-32-1
Loss Function: binary_crossentropy
|       accuracy:        0.6588000059127808     |
|       loss:            0.17940448224544525    |


Model: sequential_3-32-2
Loss Function: categorical_crossentropy
|       accuracy:        0.9872999787330627     |
|       loss:            0.04084913805127144    |


Model: sequential_3-32-2
Loss Function: poisson
|       accuracy:        0.9879000186920166     |
|       loss:            0.10404383391141891    |


Model: sequential_3-32-2
Loss Function: binary_crossentropy
|       accuracy:        0.9864000082015991     |
|       loss:            0.011171533726155758   |


Model: sequential_3-32-3
Loss Function: categorical_crossentropy
|       accuracy:        0.9884999990463257     |
|       loss:            0.03939943015575409    |


Model: sequential_3-32-3
Loss Function: poisson
|       accuracy:        0.9868000149726868     |
|       loss:            0.10447220504283905    |


Model: sequential_3-32-3
Loss Function: binary_crossentropy
|       accuracy:        0.9861999750137329     |
|       loss:            0.010479954071342945   |


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 Categorical crossentropy loss function resulted in the
following accuracy and loss:
|       accuracy:        0.9902999997138977      |
|       loss:            0.029759705066680908    |
_________________________________________________________________
 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 Categorical crossentropy loss function resulted in the
following accuracy and loss:
|       accuracy:        0.9902999997138977     |
|       loss:            0.029759705066680908   |
```