**Victor Udeh**

**CS370 Module 2**

**July 12th, 2024**

**2-2 Assignment: Identifying Hand-written Digits**

**Analysis of Parameter Changes**

Baseline Model (20 Epochs, Batch Size 128)

* Test accuracy: 94.63%

Experiment 1: Increase Number of Epochs to 30

* Test accuracy: 95.41%
* Observation: Increasing the number of epochs improved the test accuracy because the model had more opportunities to learn from the training data. This additional training time allowed the model to better fit the data, resulting in higher accuracy.

Experiment 2: Decrease Number of Epochs to 10

* Test accuracy: 92.75%
* Observation: Decreasing the number of epochs resulted in lower test accuracy, indicating that the model did not have enough time to learn effectively. With fewer epochs, the model may not have fully captured the underlying patterns in the data, leading to underfitting.

Experiment 3: Increase Batch Size to 256

* Test accuracy: 92.68%
* Observation: Increasing the batch size had a negative effect on the accuracy. Larger batch sizes can sometimes lead to more stable updates, but can also result in less frequent updates, which might not capture the data's complexity as effectively as smaller batch sizes.

Code and Results:

from \_\_future\_\_ import print\_function

import numpy as np

from keras.datasets import mnist

from keras.models import Sequential

from keras.layers import Dense, Activation, Input

from keras.optimizers import SGD

from keras.utils import to\_categorical

np.random.seed(1671) # for reproducibility

# network and training

NB\_EPOCH = 20

BATCH\_SIZE = 128

VERBOSE = 1

NB\_CLASSES = 10 # number of outputs = number of digits

OPTIMIZER = SGD() # optimizer, explained later in this chapter

N\_HIDDEN = 128

VALIDATION\_SPLIT=0.2 # how much TRAIN is reserved for VALIDATION

# data: shuffled and split between train and test sets

(X\_train, y\_train), (X\_test, y\_test) = mnist.load\_data()

RESHAPED = 784

X\_train = X\_train.reshape(60000, RESHAPED)

X\_test = X\_test.reshape(10000, RESHAPED)

X\_train = X\_train.astype('float32')

X\_test = X\_test.astype('float32')

# normalize

X\_train /= 255

X\_test /= 255

print(X\_train.shape[0], 'train samples')

print(X\_test.shape[0], 'test samples')

# convert class vectors to binary class matrices

y\_train = to\_categorical(y\_train, NB\_CLASSES)

y\_test = to\_categorical(y\_test, NB\_CLASSES)

# M\_HIDDEN hidden layers

# 10 outputs

# final stage is softmax

model = Sequential()

model.add(Input(shape=(RESHAPED,)))

model.add(Dense(N\_HIDDEN))

model.add(Activation('relu'))

model.add(Dense(N\_HIDDEN))

model.add(Activation('relu'))

model.add(Dense(NB\_CLASSES))

model.add(Activation('softmax'))

model.summary()

model.compile(loss='categorical\_crossentropy',

optimizer=OPTIMIZER,

metrics=['accuracy'])

history = model.fit(X\_train, y\_train,

batch\_size=BATCH\_SIZE, epochs=NB\_EPOCH,

verbose=VERBOSE, validation\_split=VALIDATION\_SPLIT)

score = model.evaluate(X\_test, y\_test, verbose=VERBOSE)

print("Test score:", score[0])

print('Test accuracy:', score[1])

Experiment 1: Increase Number of Epochs to 30

# Experiment 1: Increase Number of Epochs to 30

NB\_EPOCH = 30

model = Sequential()

model.add(Input(shape=(RESHAPED,)))

model.add(Dense(N\_HIDDEN))

model.add(Activation('relu'))

model.add(Dense(N\_HIDDEN))

model.add(Activation('relu'))

model.add(Dense(NB\_CLASSES))

model.add(Activation('softmax'))

model.summary()

# Create a new optimizer instance

optimizer = SGD()

model.compile(loss='categorical\_crossentropy',

optimizer=optimizer,

metrics=['accuracy'])

history = model.fit(X\_train, y\_train,

batch\_size=BATCH\_SIZE, epochs=NB\_EPOCH,

verbose=VERBOSE, validation\_split=VALIDATION\_SPLIT)

score = model.evaluate(X\_test, y\_test, verbose=VERBOSE)

print("Test score:", score[0])

print('Test accuracy:', score[1])

Experiment 2: Decrease Number of Epochs to 10

# Experiment 2: Decrease Number of Epochs to 10

NB\_EPOCH = 10

model = Sequential()

model.add(Input(shape=(RESHAPED,)))

model.add(Dense(N\_HIDDEN))

model.add(Activation('relu'))

model.add(Dense(N\_HIDDEN))

model.add(Activation('relu'))

model.add(Dense(NB\_CLASSES))

model.add(Activation('softmax'))

model.summary()

# Create a new optimizer instance

optimizer = SGD()

model.compile(loss='categorical\_crossentropy',

optimizer=optimizer,

metrics=['accuracy'])

history = model.fit(X\_train, y\_train,

batch\_size=BATCH\_SIZE, epochs=NB\_EPOCH,

verbose=VERBOSE, validation\_split=VALIDATION\_SPLIT)

score = model.evaluate(X\_test, y\_test, verbose=VERBOSE)

print("Test score:", score[0])

print('Test accuracy:', score[1])

Experiment 3: Increase Batch Size to 256

# Experiment 3: Increase Batch Size to 256

NB\_EPOCH = 20 # Resetting the number of epochs to the original value

BATCH\_SIZE = 256

model = Sequential()

model.add(Input(shape=(RESHAPED,)))

model.add(Dense(N\_HIDDEN))

model.add(Activation('relu'))

model.add(Dense(N\_HIDDEN))

model.add(Activation('relu'))

model.add(Dense(NB\_CLASSES))

model.add(Activation('softmax'))

model.summary()

# Create a new optimizer instance

optimizer = SGD()

model.compile(loss='categorical\_crossentropy',

optimizer=optimizer,

metrics=['accuracy'])

history = model.fit(X\_train, y\_train,

batch\_size=BATCH\_SIZE, epochs=NB\_EPOCH,

verbose=VERBOSE, validation\_split=VALIDATION\_SPLIT)

score = model.evaluate(X\_test, y\_test, verbose=VERBOSE)

print("Test score:", score[0])

print('Test accuracy:', score[1])

Analysis of Parameter Changes

## Analysis of Parameter Changes

### Baseline Model (20 Epochs, Batch Size 128)

- \*\*Test accuracy:\*\* 94.63%

### Experiment 1: Increase Number of Epochs to 30

- \*\*Test accuracy:\*\* 95.41%

- \*\*Observation:\*\* Increasing the number of epochs improved the test accuracy because the model had more opportunities to learn from the training data. This additional training time allowed the model to better fit the data, resulting in higher accuracy.

### Experiment 2: Decrease Number of Epochs to 10

- \*\*Test accuracy:\*\* 92.75%

- \*\*Observation:\*\* Decreasing the number of epochs resulted in lower test accuracy, indicating that the model did not have enough time to learn effectively. With fewer epochs, the model may not have fully captured the underlying patterns in the data, leading to underfitting.

### Experiment 3: Increase Batch Size to 256

- \*\*Test accuracy:\*\* 92.68%

- \*\*Observation:\*\* Increasing the batch size had a negative effect on the accuracy. Larger batch sizes can sometimes lead to more stable updates, but can also result in less frequent updates, which might not capture the data's complexity as effectively as smaller batch sizes.