Ontario Tech University

Machine Learning and Data Mining

Mini Project 3

Deadline: Tue., March 17, 11:59 PM

Instructor: Dr. Shahryar Rahnamayan, PEng, SMIEEE

Printed Digits Recognition Using ANN

Team Members	Student Numbers
Jasindan Rasalingam	100584595
Ashwin Kamalakannan	100584423

Problem

For this assignment, we were tasked with creating an artificial neural network to classify digits based on their pixel values. We had to design and train a model such that it is able to classify the digits to a reasonable accuracy and be able to test it on unlabelled data.

Design

For our implementation, we chose to use Keras, which is a TensorFlow based tool used to create neural networks. We chose to use python for quick development. For our neural net, we chose to use a sequential model with 3 layers. This means that each layer passes its outputs as the next layers inputs sequentially. For our layers we used dense layers, meaning that all neurons are fully connected. For our first 2 layers we used ReLU (rectified linear unit) activation function. This is the default activation function and is most commonly used. For our output function, we used softmax which is a probability distribution function. When compiling the model, we have to specify the cross entropy loss function and optimizer function. The loss function that we chose in this case is commonly used for numerical data sets such as ours. The optimizer function we chose is fast and has many benefits when using large data sets.

```
# Create keras model
model = tf.keras.Sequential([
    feature_layer,
    layers.Dense(128, activation='relu'),
    layers.Dense(128, activation='relu'),
    layers.Dense(10, activation='softmax')
])

# Compile Keras Model
model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy'],
)
```

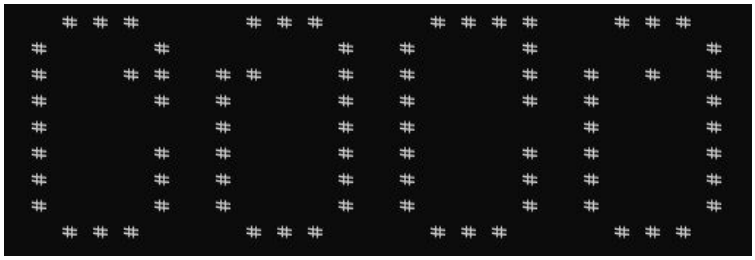
Data generation & Preparation

Data is read from a csv file, and each number is represented by 0s and 1s in an array. 0 represents an occupied pixel and 1 represents an empty one. Each number in the csv file is labelled with its corresponding value as the 46th index. We use this label to help train and validate our model. We generated 5000 digits each time for our dataset.

```
10,2,3,4,5,6,7,8,9,1,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,target
1,1,0,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,0,0,0,0,1,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,0
0,1,0,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0
1,1,0,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,0
1,1,0,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,0,0,0,0,0,1,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,0
1,1,0,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,0,0,0,0,1,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0
1,1,0,1,1,1,1,1,1,1,0,1,1,1,1,1,1,0,1,0,0,0,0,0,0,0,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0
```

We generate the data by taking our 10 pre defined digits, and adding randomly generated noise to them. Noise is generated by randomly swapping pixels from the digits. We write these to a csv file with their target values as the label.

```python
# for each number, generate 50 versions with noise
def generateDataset():
    with open('./data.csv', 'w') as d:
        with open('./numbers.csv', 'r') as f:
            reader = csv.reader(f)
            for row in reader:
                for i in range(0, 1000):
                    while(True):
                        temp = copy.copy(row)
                        a, b = random.randint(0, len(temp)-2), random.randint(0, len(temp)-2)
                        temp[a], temp[b] = temp[b], temp[a]
                        if (temp != row):
                            break
                    d.write(str(temp).replace("[", "").replace("]", "").replace("'", "").replace(" ", "") + '\n')
```



Here is an example where we have taken the digit ‘0’ and added random noise.

When preparing the data, we first split it into training, validation and testing sets. The testing set is shuffled but the others are not as it is not necessary. We use pandas to organize the datasets but we need to convert them to TensorFlow dataset objects in order to use them with our Keras model.

```
# convert datasets to TF dataset objects
batch_size = 32
train_ds = df_to_dataset(train, batch_size=batch_size)
val_ds = df_to_dataset(val, shuffle=False, batch_size=batch_size)
test_ds = df_to_dataset(test, shuffle=True, batch_size=batch_size)
```

Once the data has been converted, we create our feature layer using the labels in the first row of our csv.

```
# split data into training, validation and testing sets
train, test = train_test_split(dataframe, test_size=30)
train, val = train_test_split(train, test_size=50)
print(len(train), 'train examples')
print(len(val), 'validation examples')
print(len(test), 'test examples')

# Choose columns for feature layer
feature_columns = []
for header in ['1','2','3','4','5','6','7','8','9','10','11','12','13','
    feature_columns.append(feature_column.numeric_column(header))
feature_layer = tf.keras.layers.DenseFeatures(feature_columns)
```

Prediction Results

We ran datasets against our model for each digit 30 times.

Digit	Test Accuracy
1	0.9976
2	0.9972
3	0.9964
4	0.9975
5	0.9969
6	0.9956
7	0.9973
8	0.9965
9	0.9968
0	0.9980