Neural Networks image recognition - MultiLayer Perceptron

Use both MLNN for the following problem.

1. Add random noise (see below on `size parameter` on `np.random.normal` (https://numpy.org/doc/stable/reference/random/generated/numpy.random.normal.html)) to the images in training and testing. **Make sure each image gets a different noise feature added to it. Inspect by printing out several images. Note - the `size` parameter should match the data. **
2. Compare the `accuracy` of train and val after N epochs for MLNN with and without noise.
3. Vary the amount of noise by changing the `scale` parameter in `np.random.normal` by a factor. Use `.1, .5, 1.0, 2.0, 4.0` for the `scale` and keep track of the `accuracy` for training and validation and plot these results.

`np.random.normal`

Parameters

loc

Mean (“centre”) of the distribution.

scale

Standard deviation (spread or “width”) of the distribution. Must be non-negative.

size

Output shape. If the given shape is, e.g., (m, n, k), then m * n * k samples are drawn. If size is None (default), a single value is returned if loc and scale are both scalars. Otherwise, np.broadcast(loc, scale).size samples are drawn.

Neural Networks - Image Recognition

In [22]:

```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.optimizers import RMSprop
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
```

In [24]:
```
import matplotlib.pyplot as  plt
%matplotlib inline
```

Multi Layer Neural Network

Trains a simple deep NN on the MNIST dataset. Gets to 98.40% test accuracy after 20 epochs (there is *a lot* of margin for parameter tuning).

In [26]:
```
# the data, shuffled and split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = x_train.reshape(60000, 784)
x_test = x_test.reshape(10000, 784)
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
```

```
60000 train samples
10000 test samples
```

In [39]:
```
import numpy as np

#Use .1, .5, 1.0, 2.0, 4.0

x_train_noise10 = x_train + np.random.normal(0, 255*.10, x_train.shape
x_test_noise10 = x_test + np.random.normal(0, 255*.10, x_test.shape)

x_train_noise50 = x_train + np.random.normal(0, 255*.50, x_train.shape
x_test_noise50 = x_test + np.random.normal(0, 255*.50, x_test.shape)

x_train_noise1 = x_train + np.random.normal(0, 255*1, x_train.shape)
x_test_noise1 = x_test + np.random.normal(0, 255*1, x_test.shape)

x_train_noise2 = x_train + np.random.normal(0, 255*2, x_train.shape)
x_test_noise2 = x_test + np.random.normal(0, 255*2, x_test.shape)

x_train_noise4 = x_train + np.random.normal(0, 255*4, x_train.shape)
x_test_noise4 = x_test + np.random.normal(0, 255*4, x_test.shape)
```

In [28]:

```
batch_size = 128
num_classes = 10
epochs = 20

# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(10, activation='softmax'))

model.summary()

model.compile(loss='categorical_crossentropy',
              optimizer=RMSprop(),
              metrics=['accuracy'])

history = model.fit(x_train, y_train,
                    batch_size=batch_size,
                    epochs=epochs,
                    verbose=1,
                    validation_data=(x_test, y_test))
score_nn = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score_nn[0])
print('Test accuracy:', score_nn[1])
```

```
Model: "sequential_11"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_33 (Dense)            (None, 512)               401920

 dropout_22 (Dropout)        (None, 512)               0

 dense_34 (Dense)            (None, 512)               262656

 dropout_23 (Dropout)        (None, 512)               0

 dense_35 (Dense)            (None, 10)                5130

=================================================================
Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0
_________________________________________________________________
Epoch 1/20
469/469 [==============================] - 4s 8ms/step - loss: 0.2441
- accuracy: 0.9253 - val_loss: 0.1053 - val_accuracy: 0.9666
Epoch 2/20
469/469 [==============================] - 3s 7ms/step - loss: 0.1039
- accuracy: 0.9688 - val_loss: 0.0856 - val_accuracy: 0.9740
Epoch 3/20
```

```
469/469 [==============================] - 4s 8ms/step - loss: 0.0744
- accuracy: 0.9771 - val_loss: 0.0778 - val_accuracy: 0.9775
Epoch 4/20
469/469 [==============================] - 3s 7ms/step - loss: 0.0615
- accuracy: 0.9818 - val_loss: 0.0684 - val_accuracy: 0.9795
Epoch 5/20
469/469 [==============================] - 3s 7ms/step - loss: 0.0498
- accuracy: 0.9852 - val_loss: 0.0792 - val_accuracy: 0.9797
Epoch 6/20
469/469 [==============================] - 3s 7ms/step - loss: 0.0426
- accuracy: 0.9871 - val_loss: 0.0799 - val_accuracy: 0.9806
Epoch 7/20
469/469 [==============================] - 3s 7ms/step - loss: 0.0384
- accuracy: 0.9885 - val_loss: 0.0837 - val_accuracy: 0.9811
Epoch 8/20
469/469 [==============================] - 3s 7ms/step - loss: 0.0319
- accuracy: 0.9906 - val_loss: 0.0943 - val_accuracy: 0.9812
Epoch 9/20
469/469 [==============================] - 3s 6ms/step - loss: 0.0315
- accuracy: 0.9906 - val_loss: 0.0881 - val_accuracy: 0.9812
Epoch 10/20
469/469 [==============================] - 3s 6ms/step - loss: 0.0271
- accuracy: 0.9923 - val_loss: 0.0901 - val_accuracy: 0.9833
Epoch 11/20
469/469 [==============================] - 3s 6ms/step - loss: 0.0274
- accuracy: 0.9924 - val_loss: 0.0869 - val_accuracy: 0.9829
Epoch 12/20
469/469 [==============================] - 3s 7ms/step - loss: 0.0231
- accuracy: 0.9935 - val_loss: 0.1126 - val_accuracy: 0.9818
Epoch 13/20
469/469 [==============================] - 4s 8ms/step - loss: 0.0255
- accuracy: 0.9933 - val_loss: 0.1098 - val_accuracy: 0.9816
Epoch 14/20
469/469 [==============================] - 3s 7ms/step - loss: 0.0210
- accuracy: 0.9939 - val_loss: 0.1128 - val_accuracy: 0.9830
Epoch 15/20
469/469 [==============================] - 3s 7ms/step - loss: 0.0229
- accuracy: 0.9938 - val_loss: 0.1139 - val_accuracy: 0.9816
Epoch 16/20
469/469 [==============================] - 3s 7ms/step - loss: 0.0206
- accuracy: 0.9947 - val_loss: 0.1132 - val_accuracy: 0.9837
Epoch 17/20
469/469 [==============================] - 3s 7ms/step - loss: 0.0189
- accuracy: 0.9946 - val_loss: 0.1263 - val_accuracy: 0.9843
Epoch 18/20
469/469 [==============================] - 3s 6ms/step - loss: 0.0184
- accuracy: 0.9950 - val_loss: 0.1198 - val_accuracy: 0.9840
Epoch 19/20
469/469 [==============================] - 3s 6ms/step - loss: 0.0196
- accuracy: 0.9950 - val_loss: 0.1142 - val_accuracy: 0.9833
Epoch 20/20
469/469 [==============================] - 3s 7ms/step - loss: 0.0171
- accuracy: 0.9950 - val_loss: 0.1307 - val_accuracy: 0.9829
Test loss: 0.13065315783023834
Test accuracy: 0.9829000234603882
```

In [29]:

```
batch_size = 128
num_classes = 10
epochs = 20

model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(10, activation='softmax'))

model.summary()

model.compile(loss='categorical_crossentropy',
              optimizer=RMSprop(),
              metrics=['accuracy'])

history = model.fit(x_train_noise10, y_train,
                    batch_size=batch_size,
                    epochs=epochs,
                    verbose=1,
                    validation_data=(x_test_noise10, y_test))
score_nn10 = model.evaluate(x_test_noise10, y_test, verbose=0)
print('Test loss:', score_nn10[0])
print('Test accuracy:', score_nn10[1])
```

```
Model: "sequential_12"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_36 (Dense)            (None, 512)               401920

 dropout_24 (Dropout)        (None, 512)               0

 dense_37 (Dense)            (None, 512)               262656

 dropout_25 (Dropout)        (None, 512)               0

 dense_38 (Dense)            (None, 10)                5130

=================================================================
Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0
_________________________________________________________________
Epoch 1/20
469/469 [==============================] - 4s 8ms/step - loss: 3.8668
- accuracy: 0.1072 - val_loss: 2.3022 - val_accuracy: 0.1135
Epoch 2/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3051
- accuracy: 0.1165 - val_loss: 2.3078 - val_accuracy: 0.1126
Epoch 3/20
469/469 [==============================] - 3s 6ms/step - loss: 2.2933
- accuracy: 0.1206 - val_loss: 2.3033 - val_accuracy: 0.1123
Epoch 4/20
469/469 [==============================] - 3s 6ms/step - loss: 2.2823
```

```
- accuracy: 0.1255 - val_loss: 2.3042 - val_accuracy: 0.1130
Epoch 5/20
469/469 [==============================] - 3s 7ms/step - loss: 2.2740
- accuracy: 0.1287 - val_loss: 2.3033 - val_accuracy: 0.1134
Epoch 6/20
469/469 [==============================] - 3s 6ms/step - loss: 2.2692
- accuracy: 0.1314 - val_loss: 2.3021 - val_accuracy: 0.1132
Epoch 7/20
469/469 [==============================] - 3s 6ms/step - loss: 2.2706
- accuracy: 0.1325 - val_loss: 2.3016 - val_accuracy: 0.1137
Epoch 8/20
469/469 [==============================] - 3s 6ms/step - loss: 2.2660
- accuracy: 0.1342 - val_loss: 2.3051 - val_accuracy: 0.1133
Epoch 9/20
469/469 [==============================] - 3s 6ms/step - loss: 2.2633
- accuracy: 0.1357 - val_loss: 2.3048 - val_accuracy: 0.1135
Epoch 10/20
469/469 [==============================] - 3s 6ms/step - loss: 2.2620
- accuracy: 0.1371 - val_loss: 2.3045 - val_accuracy: 0.1135
Epoch 11/20
469/469 [==============================] - 3s 6ms/step - loss: 2.2607
- accuracy: 0.1388 - val_loss: 2.3043 - val_accuracy: 0.1138
Epoch 12/20
469/469 [==============================] - 3s 7ms/step - loss: 2.2604
- accuracy: 0.1399 - val_loss: 2.3051 - val_accuracy: 0.1133
Epoch 13/20
469/469 [==============================] - 3s 6ms/step - loss: 2.2595
- accuracy: 0.1402 - val_loss: 2.3046 - val_accuracy: 0.1134
Epoch 14/20
469/469 [==============================] - 3s 6ms/step - loss: 2.2540
- accuracy: 0.1417 - val_loss: 2.3055 - val_accuracy: 0.1127
Epoch 15/20
469/469 [==============================] - 3s 6ms/step - loss: 2.2552
- accuracy: 0.1420 - val_loss: 2.3056 - val_accuracy: 0.1137
Epoch 16/20
469/469 [==============================] - 3s 6ms/step - loss: 2.2531
- accuracy: 0.1430 - val_loss: 2.3041 - val_accuracy: 0.1131
Epoch 17/20
469/469 [==============================] - 3s 6ms/step - loss: 2.2556
- accuracy: 0.1431 - val_loss: 2.3023 - val_accuracy: 0.1134
Epoch 18/20
469/469 [==============================] - 3s 6ms/step - loss: 2.2508
- accuracy: 0.1454 - val_loss: 2.3048 - val_accuracy: 0.1137
Epoch 19/20
469/469 [==============================] - 3s 7ms/step - loss: 2.2498
- accuracy: 0.1439 - val_loss: 2.3028 - val_accuracy: 0.1133
Epoch 20/20
469/469 [==============================] - 3s 6ms/step - loss: 2.2507
- accuracy: 0.1447 - val_loss: 2.3046 - val_accuracy: 0.1135
Test loss: 2.3046178817749023
Test accuracy: 0.11349999904632568
```

In [32]:
```
batch_size = 128
num_classes = 10
epochs = 20

model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(10, activation='softmax'))

model.summary()

model.compile(loss='categorical_crossentropy',
              optimizer=RMSprop(),
              metrics=['accuracy'])

history = model.fit(x_train_noise50, y_train,
                    batch_size=batch_size,
                    epochs=epochs,
                    verbose=1,
                    validation_data=(x_test_noise50, y_test))
score_nn50 = model.evaluate(x_test_noise50, y_test, verbose=0)
print('Test loss:', score_nn50[0])
print('Test accuracy:', score_nn50[1])
```

```
Model: "sequential_15"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_45 (Dense)            (None, 512)               401920

 dropout_30 (Dropout)        (None, 512)               0

 dense_46 (Dense)            (None, 512)               262656

 dropout_31 (Dropout)        (None, 512)               0

 dense_47 (Dense)            (None, 10)                5130

=================================================================
Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0
_________________________________________________________________
Epoch 1/20
469/469 [==============================] - 4s 8ms/step - loss: 12.186
6 - accuracy: 0.1081 - val_loss: 2.3023 - val_accuracy: 0.1137
Epoch 2/20
469/469 [==============================] - 3s 7ms/step - loss: 2.3130
- accuracy: 0.1136 - val_loss: 2.3017 - val_accuracy: 0.1136
Epoch 3/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3124
- accuracy: 0.1142 - val_loss: 2.3025 - val_accuracy: 0.1135
Epoch 4/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3099
```

```
- accuracy: 0.1148 - val_loss: 2.3017 - val_accuracy: 0.1135
Epoch 5/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3065
- accuracy: 0.1152 - val_loss: 2.3027 - val_accuracy: 0.1136
Epoch 6/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3090
- accuracy: 0.1157 - val_loss: 2.3019 - val_accuracy: 0.1135
Epoch 7/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3041
- accuracy: 0.1160 - val_loss: 2.3014 - val_accuracy: 0.1134
Epoch 8/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3050
- accuracy: 0.1159 - val_loss: 2.3015 - val_accuracy: 0.1134
Epoch 9/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3010
- accuracy: 0.1163 - val_loss: 2.3011 - val_accuracy: 0.1135
Epoch 10/20
469/469 [==============================] - 3s 6ms/step - loss: 2.2996
- accuracy: 0.1168 - val_loss: 2.3009 - val_accuracy: 0.1135
Epoch 11/20
469/469 [==============================] - 3s 6ms/step - loss: 2.2991
- accuracy: 0.1164 - val_loss: 2.3011 - val_accuracy: 0.1135
Epoch 12/20
469/469 [==============================] - 3s 6ms/step - loss: 2.2967
- accuracy: 0.1168 - val_loss: 2.3010 - val_accuracy: 0.1137
Epoch 13/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3010
- accuracy: 0.1168 - val_loss: 2.3013 - val_accuracy: 0.1136
Epoch 14/20
469/469 [==============================] - 3s 6ms/step - loss: 2.2986
- accuracy: 0.1171 - val_loss: 2.3011 - val_accuracy: 0.1135
Epoch 15/20
469/469 [==============================] - 3s 7ms/step - loss: 2.2990
- accuracy: 0.1171 - val_loss: 2.3010 - val_accuracy: 0.1135
Epoch 16/20
469/469 [==============================] - 3s 6ms/step - loss: 2.2960
- accuracy: 0.1176 - val_loss: 2.3016 - val_accuracy: 0.1136
Epoch 17/20
469/469 [==============================] - 3s 7ms/step - loss: 2.2936
- accuracy: 0.1177 - val_loss: 2.3011 - val_accuracy: 0.1135
Epoch 18/20
469/469 [==============================] - 3s 6ms/step - loss: 2.2953
- accuracy: 0.1176 - val_loss: 2.3011 - val_accuracy: 0.1135
Epoch 19/20
469/469 [==============================] - 3s 6ms/step - loss: 2.2992
- accuracy: 0.1176 - val_loss: 2.3034 - val_accuracy: 0.1135
Epoch 20/20
469/469 [==============================] - 3s 6ms/step - loss: 2.2955
- accuracy: 0.1176 - val_loss: 2.3009 - val_accuracy: 0.1135
Test loss: 2.3009188175201416
Test accuracy: 0.11349999904632568
```

In [40]:

```
batch_size = 128
num_classes = 10
epochs = 20

model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(10, activation='softmax'))

model.summary()

model.compile(loss='categorical_crossentropy',
              optimizer=RMSprop(),
              metrics=['accuracy'])

history = model.fit(x_train_noise1, y_train,
                    batch_size=batch_size,
                    epochs=epochs,
                    verbose=1,
                    validation_data=(x_test_noise50, y_test))
score_nn1 = model.evaluate(x_test_noise1, y_test, verbose=0)
print('Test loss:', score_nn1[0])
print('Test accuracy:', score_nn1[1])
```

```
Model: "sequential_16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense_48 (Dense)            (None, 512)               401920    
                                                                 
 dropout_32 (Dropout)        (None, 512)               0         
                                                                 
 dense_49 (Dense)            (None, 512)               262656    
                                                                 
 dropout_33 (Dropout)        (None, 512)               0         
                                                                 
 dense_50 (Dense)            (None, 10)                5130      
                                                                 
=================================================================
Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0
_________________________________________________________________
Epoch 1/20
469/469 [==============================] - 4s 8ms/step - loss: 21.647
6 - accuracy: 0.1103 - val_loss: 2.3032 - val_accuracy: 0.1134
Epoch 2/20
469/469 [==============================] - 3s 7ms/step - loss: 2.3326
- accuracy: 0.1131 - val_loss: 2.3010 - val_accuracy: 0.1134
Epoch 3/20
469/469 [==============================] - 4s 8ms/step - loss: 2.3199
- accuracy: 0.1137 - val_loss: 2.3013 - val_accuracy: 0.1133
Epoch 4/20
469/469 [==============================] - 4s 7ms/step - loss: 2.3186
```

```
- accuracy: 0.1141 - val_loss: 2.3024 - val_accuracy: 0.1134
Epoch 5/20
469/469 [==============================] - 3s 7ms/step - loss: 2.3162
- accuracy: 0.1145 - val_loss: 2.3016 - val_accuracy: 0.1135
Epoch 6/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3127
- accuracy: 0.1147 - val_loss: 2.3012 - val_accuracy: 0.1134
Epoch 7/20
469/469 [==============================] - 3s 7ms/step - loss: 2.3106
- accuracy: 0.1148 - val_loss: 2.3010 - val_accuracy: 0.1135
Epoch 8/20
469/469 [==============================] - 4s 8ms/step - loss: 2.3088
- accuracy: 0.1148 - val_loss: 2.3015 - val_accuracy: 0.1135
Epoch 9/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3051
- accuracy: 0.1151 - val_loss: 2.3011 - val_accuracy: 0.1135
Epoch 10/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3043
- accuracy: 0.1151 - val_loss: 2.3010 - val_accuracy: 0.1135
Epoch 11/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3007
- accuracy: 0.1153 - val_loss: 2.3010 - val_accuracy: 0.1135
Epoch 12/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3068
- accuracy: 0.1154 - val_loss: 2.3010 - val_accuracy: 0.1135
Epoch 13/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3032
- accuracy: 0.1157 - val_loss: 2.3011 - val_accuracy: 0.1135
Epoch 14/20
469/469 [==============================] - 3s 7ms/step - loss: 2.3063
- accuracy: 0.1157 - val_loss: 2.3023 - val_accuracy: 0.1135
Epoch 15/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3037
- accuracy: 0.1156 - val_loss: 2.3014 - val_accuracy: 0.1135
Epoch 16/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3031
- accuracy: 0.1156 - val_loss: 2.3010 - val_accuracy: 0.1135
Epoch 17/20
469/469 [==============================] - 3s 7ms/step - loss: 2.3043
- accuracy: 0.1159 - val_loss: 2.3011 - val_accuracy: 0.1135
Epoch 18/20
469/469 [==============================] - 3s 7ms/step - loss: 2.3053
- accuracy: 0.1158 - val_loss: 2.3012 - val_accuracy: 0.1135
Epoch 19/20
469/469 [==============================] - 3s 6ms/step - loss: 2.2993
- accuracy: 0.1158 - val_loss: 2.3010 - val_accuracy: 0.1135
Epoch 20/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3016
- accuracy: 0.1158 - val_loss: 2.3010 - val_accuracy: 0.1135
Test loss: 2.300942897796631
Test accuracy: 0.1136000007390976
```

In [41]:

```
batch_size = 128
num_classes = 10
epochs = 20

model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(10, activation='softmax'))

model.summary()

model.compile(loss='categorical_crossentropy',
              optimizer=RMSprop(),
              metrics=['accuracy'])

history = model.fit(x_train_noise2, y_train,
                    batch_size=batch_size,
                    epochs=epochs,
                    verbose=1,
                    validation_data=(x_test_noise2, y_test))
score_nn2 = model.evaluate(x_test_noise2, y_test, verbose=0)
print('Test loss:', score_nn2[0])
print('Test accuracy:', score_nn2[1])
```

```
Model: "sequential_17"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_51 (Dense)            (None, 512)               401920

 dropout_34 (Dropout)        (None, 512)               0

 dense_52 (Dense)            (None, 512)               262656

 dropout_35 (Dropout)        (None, 512)               0

 dense_53 (Dense)            (None, 10)                5130

=================================================================
Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0
_________________________________________________________________
Epoch 1/20
469/469 [==============================] - 4s 8ms/step - loss: 36.621
2 - accuracy: 0.1083 - val_loss: 2.3022 - val_accuracy: 0.1135
Epoch 2/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3491
- accuracy: 0.1128 - val_loss: 2.3014 - val_accuracy: 0.1133
Epoch 3/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3278
- accuracy: 0.1129 - val_loss: 2.3011 - val_accuracy: 0.1135
Epoch 4/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3256
```

```
- accuracy: 0.1132 - val_loss: 2.3011 - val_accuracy: 0.1135
Epoch 5/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3203
- accuracy: 0.1135 - val_loss: 2.3010 - val_accuracy: 0.1135
Epoch 6/20
469/469 [==============================] - 3s 7ms/step - loss: 2.3225
- accuracy: 0.1136 - val_loss: 2.3010 - val_accuracy: 0.1135
Epoch 7/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3155
- accuracy: 0.1136 - val_loss: 2.3010 - val_accuracy: 0.1135
Epoch 8/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3112
- accuracy: 0.1136 - val_loss: 2.3010 - val_accuracy: 0.1135
Epoch 9/20
469/469 [==============================] - 3s 7ms/step - loss: 2.3114
- accuracy: 0.1137 - val_loss: 2.3009 - val_accuracy: 0.1135
Epoch 10/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3112
- accuracy: 0.1138 - val_loss: 2.3012 - val_accuracy: 0.1135
Epoch 11/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3111
- accuracy: 0.1138 - val_loss: 2.3010 - val_accuracy: 0.1135
Epoch 12/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3155
- accuracy: 0.1140 - val_loss: 2.3011 - val_accuracy: 0.1135
Epoch 13/20
469/469 [==============================] - 3s 7ms/step - loss: 2.3085
- accuracy: 0.1140 - val_loss: 2.3010 - val_accuracy: 0.1135
Epoch 14/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3061
- accuracy: 0.1142 - val_loss: 2.3009 - val_accuracy: 0.1135
Epoch 15/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3065
- accuracy: 0.1143 - val_loss: 2.3010 - val_accuracy: 0.1135
Epoch 16/20
469/469 [==============================] - 3s 7ms/step - loss: 2.3008
- accuracy: 0.1142 - val_loss: 2.3010 - val_accuracy: 0.1135
Epoch 17/20
469/469 [==============================] - 3s 7ms/step - loss: 2.3048
- accuracy: 0.1143 - val_loss: 2.3010 - val_accuracy: 0.1135
Epoch 18/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3009
- accuracy: 0.1143 - val_loss: 2.3010 - val_accuracy: 0.1135
Epoch 19/20
469/469 [==============================] - 3s 7ms/step - loss: 2.3053
- accuracy: 0.1145 - val_loss: 2.3010 - val_accuracy: 0.1135
Epoch 20/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3013
- accuracy: 0.1144 - val_loss: 2.3010 - val_accuracy: 0.1135
Test loss: 2.3010334968566895
Test accuracy: 0.11349999904632568
```

In [42]:

```
batch_size = 128
num_classes = 10
epochs = 20

model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(10, activation='softmax'))

model.summary()

model.compile(loss='categorical_crossentropy',
              optimizer=RMSprop(),
              metrics=['accuracy'])

history = model.fit(x_train_noise4, y_train,
                    batch_size=batch_size,
                    epochs=epochs,
                    verbose=1,
                    validation_data=(x_test_noise4, y_test))
score_nn4 = model.evaluate(x_test_noise4, y_test, verbose=0)
print('Test loss:', score_nn4[0])
print('Test accuracy:', score_nn4[1])
```

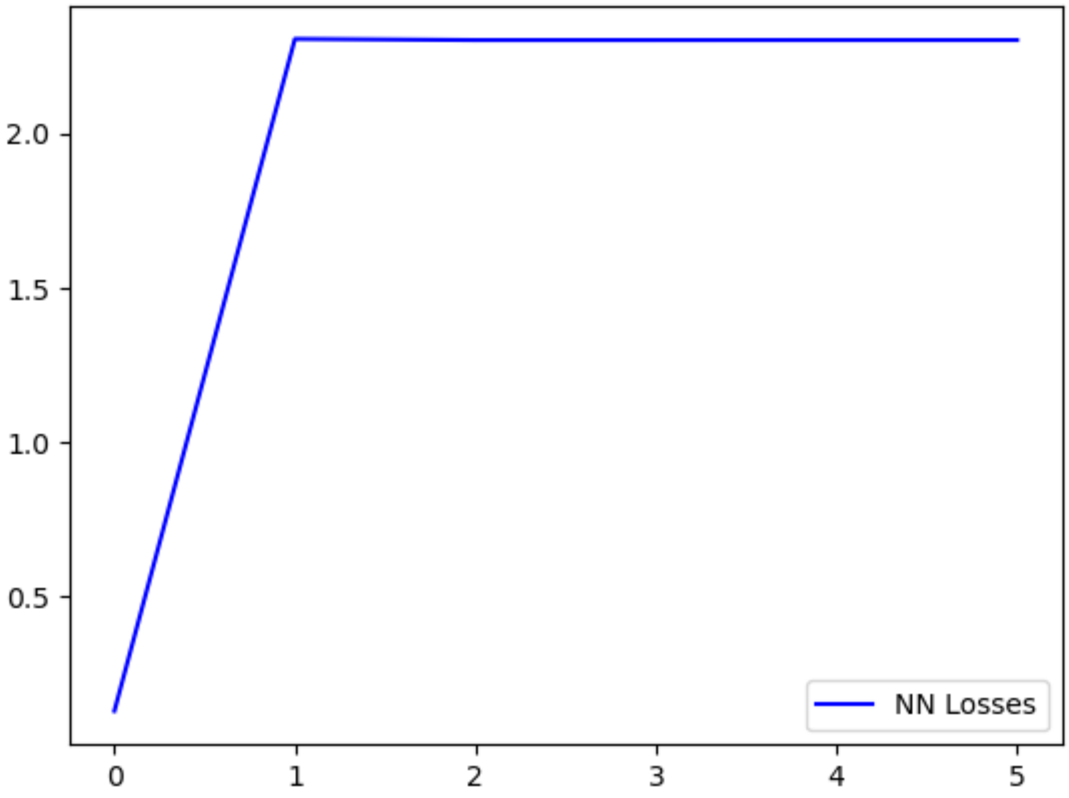
```
Model: "sequential_18"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_54 (Dense)            (None, 512)               401920

 dropout_36 (Dropout)        (None, 512)               0

 dense_55 (Dense)            (None, 512)               262656

 dropout_37 (Dropout)        (None, 512)               0

 dense_56 (Dense)            (None, 10)                5130

=================================================================
Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0
_________________________________________________________________
Epoch 1/20
469/469 [==============================] - 4s 8ms/step - loss: 76.591
8 - accuracy: 0.1070 - val_loss: 2.3032 - val_accuracy: 0.1135
Epoch 2/20
469/469 [==============================] - 3s 7ms/step - loss: 2.3895
- accuracy: 0.1125 - val_loss: 2.3014 - val_accuracy: 0.1135
Epoch 3/20
469/469 [==============================] - 3s 7ms/step - loss: 2.3526
- accuracy: 0.1127 - val_loss: 2.3043 - val_accuracy: 0.1134
Epoch 4/20
469/469 [==============================] - 3s 7ms/step - loss: 2.3438
```

```
- accuracy: 0.1130 - val_loss: 2.3010 - val_accuracy: 0.1135
Epoch 5/20
469/469 [==============================] - 4s 8ms/step - loss: 2.3303
- accuracy: 0.1129 - val_loss: 2.3030 - val_accuracy: 0.1135
Epoch 6/20
469/469 [==============================] - 4s 8ms/step - loss: 2.3284
- accuracy: 0.1129 - val_loss: 2.3014 - val_accuracy: 0.1134
Epoch 7/20
469/469 [==============================] - 3s 7ms/step - loss: 2.3214
- accuracy: 0.1131 - val_loss: 2.3023 - val_accuracy: 0.1134
Epoch 8/20
469/469 [==============================] - 3s 7ms/step - loss: 2.3215
- accuracy: 0.1132 - val_loss: 2.3010 - val_accuracy: 0.1135
Epoch 9/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3171
- accuracy: 0.1133 - val_loss: 2.3010 - val_accuracy: 0.1135
Epoch 10/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3147
- accuracy: 0.1134 - val_loss: 2.3011 - val_accuracy: 0.1135
Epoch 11/20
469/469 [==============================] - 3s 7ms/step - loss: 2.3187
- accuracy: 0.1132 - val_loss: 2.3010 - val_accuracy: 0.1135
Epoch 12/20
469/469 [==============================] - 3s 7ms/step - loss: 2.3192
- accuracy: 0.1133 - val_loss: 2.3010 - val_accuracy: 0.1135
Epoch 13/20
469/469 [==============================] - 3s 7ms/step - loss: 2.3122
- accuracy: 0.1132 - val_loss: 2.3011 - val_accuracy: 0.1135
Epoch 14/20
469/469 [==============================] - 3s 7ms/step - loss: 2.3130
- accuracy: 0.1133 - val_loss: 2.3010 - val_accuracy: 0.1135
Epoch 15/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3113
- accuracy: 0.1135 - val_loss: 2.3010 - val_accuracy: 0.1135
Epoch 16/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3057
- accuracy: 0.1135 - val_loss: 2.3010 - val_accuracy: 0.1135
Epoch 17/20
469/469 [==============================] - 3s 7ms/step - loss: 2.3083
- accuracy: 0.1135 - val_loss: 2.3013 - val_accuracy: 0.1135
Epoch 18/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3125
- accuracy: 0.1136 - val_loss: 2.3010 - val_accuracy: 0.1135
Epoch 19/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3084
- accuracy: 0.1136 - val_loss: 2.3010 - val_accuracy: 0.1135
Epoch 20/20
469/469 [==============================] - 3s 6ms/step - loss: 2.3079
- accuracy: 0.1135 - val_loss: 2.3010 - val_accuracy: 0.1135
Test loss: 2.3010404109954834
Test accuracy: 0.11349999904632568
```

Visualization of Loss

In [43]:
```
nn_losses = np.array([score_nn[0], score_nn10[0], score_nn50[0], score
plt.plot(nn_losses, label='NN Losses', color='b')
plt.legend()
```

Out[43]:
```
<matplotlib.legend.Legend at 0x7fe36dc80a00>
```



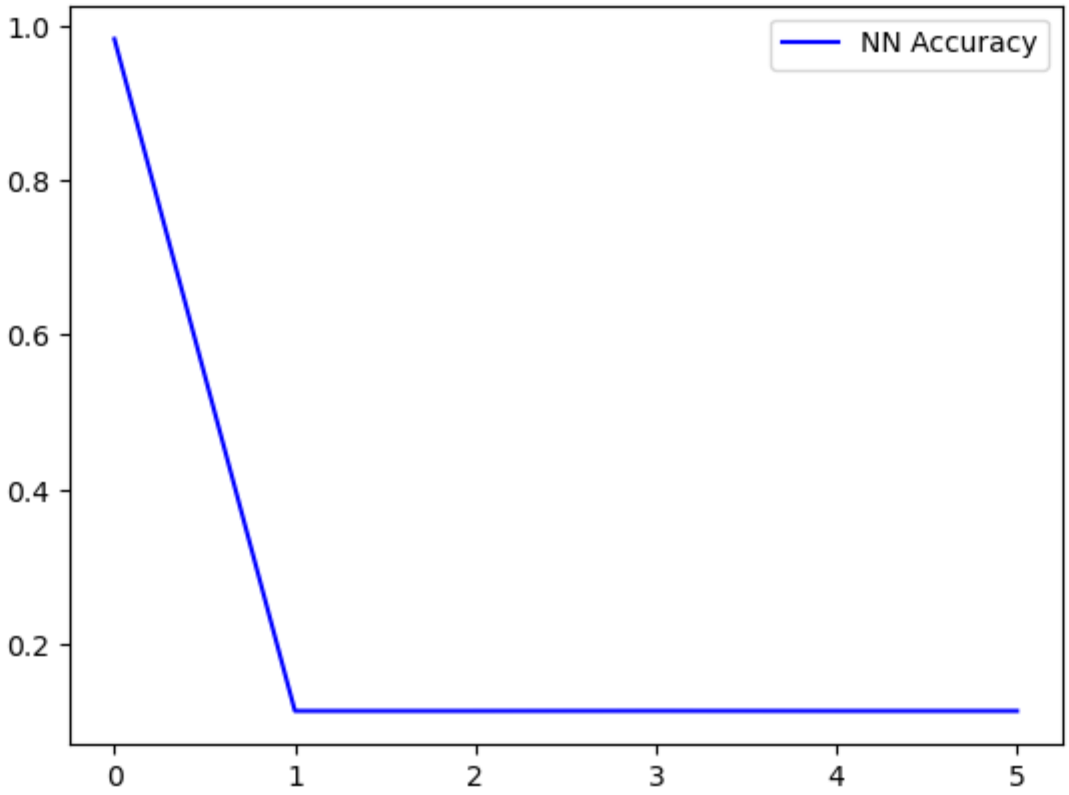
In [44]:
```
nn_losses
```

Out[44]:
```
array([0.13065316, 2.30461788, 2.30091882, 2.3009429 , 2.3010335 ,
       2.30104041])
```

Visualization of Accuracy

```
In [46]: nn_accur = np.array([score_nn[1], score_nn10[1], score_nn50[1], score_
         plt.plot(nn_accur, label='NN Accuracy', color='b')
         plt.legend()
```

```
Out[46]: <matplotlib.legend.Legend at 0x7fe3193e4df0>
```



```
In [47]: nn_accur
```

```
Out[47]: array([0.98290002, 0.1135    , 0.1135    , 0.1136    , 0.1135    ,
                0.1135    ])
```

As seen in the visualizations of both loss and accuracy above, adding noise (.10 to .50 to 1 to 2 to 4) greatly increases loss and decreases accuracy- even at a low level. It is interesting though that once noise wrecks the result, adding more noise does not further degrade accuracy or increase loss.

```
In [ ]:
```