Neural Networks image recognition - ConvNet

1. Add random noise (see below on `size parameter` on `np.random.normal`) 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.
4. Compare these results with the previous week where we used a MultiLayer Perceptron (this week we use a ConvNet).

Neural Networks - Image Recognition

In [63]:
```
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
from keras import backend
```

In [64]:
```
import matplotlib.pyplot as  plt
%matplotlib inline
import numpy as np
```

Conv Net

Trains a simple convnet on the MNIST dataset. Gets to 99.25% test accuracy after 12 epochs (there is still a lot of margin for parameter tuning).

In [101…
```
# input image dimensions
img_rows, img_cols = 28, 28

scale = [0, .1, .5, 1.0, 2.0, 4.0]
batch_size = 128
num_classes = 10
epochs = 12

losses = []
acc = []
```

In [102…
```
for s in scale:
    # the data, shuffled and split between train and test sets
    (x_train, y_train), (x_test, y_test) = mnist.load_data()

    # 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()

    if backend.image_data_format() == 'channels_first':
        x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
        x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
        input_shape = (1, img_rows, img_cols)
    else:
        x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
        x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
        input_shape = (img_rows, img_cols, 1)

    x_train = x_train.astype('float32')
    x_test = x_test.astype('float32')
    x_train /= 255
    x_test /= 255

    new_train = x_train + np.random.normal(0, s, size=(60000, 28, 28, 1))
    new_test = x_test + np.random.normal(0, s, size=(10000, 28, 28, 1))
    print('new_train shape:', new_train.shape)
    print('new_test shape:', new_test.shape)

    model.add(Conv2D(32, kernel_size=(3, 3),
                     activation='relu',
                     input_shape=input_shape))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))
    model.add(Flatten())
```

```
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(num_classes, activation='softmax'))

    model.compile(loss=keras.losses.categorical_crossentropy,
                  optimizer=keras.optimizers.Adadelta(),
                  metrics=['accuracy'])

    model.fit(new_train, y_train,
              batch_size=batch_size,
              epochs=epochs,
              verbose=1,
              validation_data=(new_test, y_test))
    score = model.evaluate(new_test, y_test, verbose=0)
    losses.append(score[0])
    acc.append(score[1])
    print('Test loss:', score[0])
    print('Test accuracy:', score[1])
```

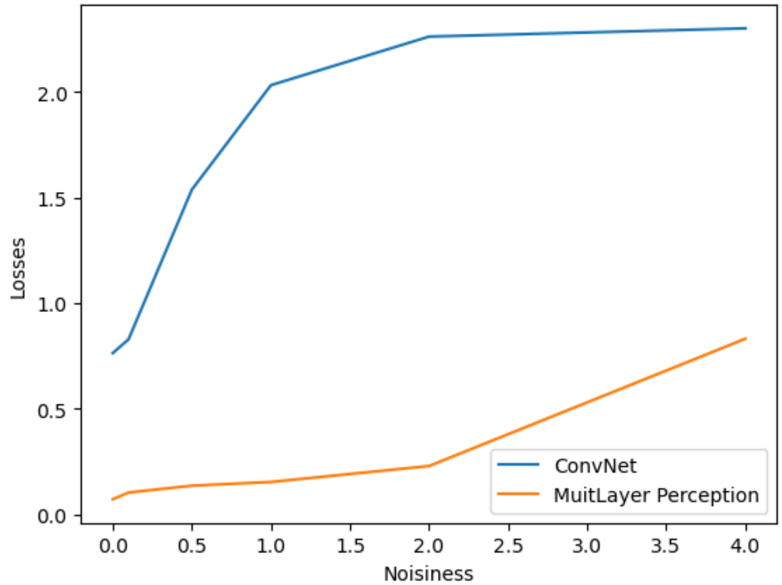
```
new_train shape: (60000, 28, 28, 1)
new_test shape: (10000, 28, 28, 1)
Epoch 1/12
469/469 [==============================] - 43s 85ms/step - loss: 2.2817 - accuracy: 0.1517 - val_loss: 2.2442 - val_accuracy: 0.35
32
Epoch 2/12
469/469 [==============================] - 42s 90ms/step - loss: 2.2259 - accuracy: 0.2674 - val_loss: 2.1766 - val_accuracy: 0.52
28
Epoch 3/12
469/469 [==============================] - 44s 94ms/step - loss: 2.1565 - accuracy: 0.3756 - val_loss: 2.0899 - val_accuracy: 0.61
90
Epoch 4/12
469/469 [==============================] - 44s 94ms/step - loss: 2.0671 - accuracy: 0.4588 - val_loss: 1.9754 - val_accuracy: 0.68
06
Epoch 5/12
469/469 [==============================] - 44s 93ms/step - loss: 1.9494 - accuracy: 0.5142 - val_loss: 1.8269 - val_accuracy: 0.71
76
Epoch 6/12
469/469 [==============================] - 40s 85ms/step - loss: 1.8055 - accuracy: 0.5545 - val_loss: 1.6462 - val_accuracy: 0.74
39
Epoch 7/12
469/469 [==============================] - 43s 91ms/step - loss: 1.6451 - accuracy: 0.5864 - val_loss: 1.4491 - val_accuracy: 0.76
86
Epoch 8/12
469/469 [==============================] - 43s 92ms/step - loss: 1.4797 - accuracy: 0.6171 - val_loss: 1.2567 - val_accuracy: 0.78
88
Epoch 9/12
469/469 [==============================] - 45s 96ms/step - loss: 1.3283 - accuracy: 0.6435 - val_loss: 1.0872 - val_accuracy: 0.80
68
Epoch 10/12
469/469 [==============================] - 42s 90ms/step - loss: 1.1989 - accuracy: 0.6698 - val_loss: 0.9508 - val_accuracy: 0.81
78
Epoch 11/12
469/469 [==============================] - 44s 94ms/step - loss: 1.1032 - accuracy: 0.6857 - val_loss: 0.8454 - val_accuracy: 0.82
77
Epoch 12/12
469/469 [==============================] - 44s 93ms/step - loss: 1.0208 - accuracy: 0.7027 - val_loss: 0.7642 - val_accuracy: 0.83
49
Test loss: 0.764194667339325
Test accuracy: 0.8349000215530396
new_train shape: (60000, 28, 28, 1)
new_test shape: (10000, 28, 28, 1)
Epoch 1/12
469/469 [==============================] - 46s 91ms/step - loss: 2.2888 - accuracy: 0.1427 - val_loss: 2.2531 - val_accuracy: 0.37
99
Epoch 2/12
469/469 [==============================] - 44s 93ms/step - loss: 2.2370 - accuracy: 0.2409 - val_loss: 2.1906 - val_accuracy: 0.49
14
Epoch 3/12
469/469 [==============================] - 41s 88ms/step - loss: 2.1754 - accuracy: 0.3308 - val_loss: 2.1136 - val_accuracy: 0.55
03
Epoch 4/12
469/469 [==============================] - 44s 93ms/step - loss: 2.0972 - accuracy: 0.4006 - val_loss: 2.0134 - val_accuracy: 0.61
75
Epoch 5/12
469/469 [==============================] - 43s 92ms/step - loss: 1.9968 - accuracy: 0.4581 - val_loss: 1.8843 - val_accuracy: 0.67
54
Epoch 6/12
469/469 [==============================] - 41s 87ms/step - loss: 1.8717 - accuracy: 0.5092 - val_loss: 1.7259 - val_accuracy: 0.71
75
Epoch 7/12
469/469 [==============================] - 44s 94ms/step - loss: 1.7273 - accuracy: 0.5480 - val_loss: 1.5458 - val_accuracy: 0.74
60
Epoch 8/12
469/469 [==============================] - 44s 93ms/step - loss: 1.5741 - accuracy: 0.5833 - val_loss: 1.3610 - val_accuracy: 0.76
45
Epoch 9/12
469/469 [==============================] - 41s 88ms/step - loss: 1.4274 - accuracy: 0.6124 - val_loss: 1.1893 - val_accuracy: 0.78
02
Epoch 10/12
469/469 [==============================] - 44s 94ms/step - loss: 1.2996 - accuracy: 0.6358 - val_loss: 1.0420 - val_accuracy: 0.79
68
Epoch 11/12
469/469 [==============================] - 43s 92ms/step - loss: 1.1887 - accuracy: 0.6600 - val_loss: 0.9226 - val_accuracy: 0.81
08
Epoch 12/12
469/469 [==============================] - 41s 87ms/step - loss: 1.0986 - accuracy: 0.6803 - val_loss: 0.8296 - val_accuracy: 0.82
02
Test loss: 0.8295845985412598
Test accuracy: 0.8202000260353088
new_train shape: (60000, 28, 28, 1)
new_test shape: (10000, 28, 28, 1)
Epoch 1/12
469/469 [==============================] - 46s 90ms/step - loss: 2.3075 - accuracy: 0.1119 - val_loss: 2.2786 - val_accuracy: 0.19
93
Epoch 2/12
```

```
469/469 [==============================] - 42s 89ms/step - loss: 2.2777 - accuracy: 0.1461 - val_loss: 2.2489 - val_accuracy: 0.31
32
Epoch 3/12
469/469 [==============================] - 38s 81ms/step - loss: 2.2515 - accuracy: 0.1877 - val_loss: 2.2185 - val_accuracy: 0.39
82
Epoch 4/12
469/469 [==============================] - 40s 85ms/step - loss: 2.2228 - accuracy: 0.2219 - val_loss: 2.1822 - val_accuracy: 0.47
69
Epoch 5/12
469/469 [==============================] - 42s 90ms/step - loss: 2.1882 - accuracy: 0.2569 - val_loss: 2.1377 - val_accuracy: 0.53
05
Epoch 6/12
469/469 [==============================] - 43s 91ms/step - loss: 2.1465 - accuracy: 0.2963 - val_loss: 2.0844 - val_accuracy: 0.57
00
Epoch 7/12
469/469 [==============================] - 39s 83ms/step - loss: 2.0950 - accuracy: 0.3327 - val_loss: 2.0215 - val_accuracy: 0.60
86
Epoch 8/12
469/469 [==============================] - 43s 91ms/step - loss: 2.0371 - accuracy: 0.3684 - val_loss: 1.9470 - val_accuracy: 0.63
53
Epoch 9/12
469/469 [==============================] - 42s 89ms/step - loss: 1.9668 - accuracy: 0.4081 - val_loss: 1.8598 - val_accuracy: 0.66
48
Epoch 10/12
469/469 [==============================] - 48s 102ms/step - loss: 1.8857 - accuracy: 0.4433 - val_loss: 1.7608 - val_accuracy: 0.6
890
Epoch 11/12
469/469 [==============================] - 44s 95ms/step - loss: 1.7999 - accuracy: 0.4765 - val_loss: 1.6523 - val_accuracy: 0.71
19
Epoch 12/12
469/469 [==============================] - 44s 93ms/step - loss: 1.7019 - accuracy: 0.5086 - val_loss: 1.5383 - val_accuracy: 0.72
59
Test loss: 1.538321852684021
Test accuracy: 0.7258999943733215
new_train shape: (60000, 28, 28, 1)
new_test shape: (10000, 28, 28, 1)
Epoch 1/12
469/469 [==============================] - 48s 96ms/step - loss: 2.3319 - accuracy: 0.1089 - val_loss: 2.2892 - val_accuracy: 0.13
64
Epoch 2/12
469/469 [==============================] - 44s 94ms/step - loss: 2.3043 - accuracy: 0.1195 - val_loss: 2.2759 - val_accuracy: 0.18
04
Epoch 3/12
469/469 [==============================] - 43s 92ms/step - loss: 2.2904 - accuracy: 0.1304 - val_loss: 2.2651 - val_accuracy: 0.21
98
Epoch 4/12
469/469 [==============================] - 42s 89ms/step - loss: 2.2775 - accuracy: 0.1447 - val_loss: 2.2524 - val_accuracy: 0.25
34
Epoch 5/12
469/469 [==============================] - 44s 94ms/step - loss: 2.2674 - accuracy: 0.1539 - val_loss: 2.2368 - val_accuracy: 0.29
17
Epoch 6/12
469/469 [==============================] - 43s 91ms/step - loss: 2.2545 - accuracy: 0.1674 - val_loss: 2.2188 - val_accuracy: 0.32
76
Epoch 7/12
469/469 [==============================] - 42s 90ms/step - loss: 2.2401 - accuracy: 0.1798 - val_loss: 2.1970 - val_accuracy: 0.36
17
Epoch 8/12
469/469 [==============================] - 44s 93ms/step - loss: 2.2223 - accuracy: 0.1938 - val_loss: 2.1718 - val_accuracy: 0.39
65
Epoch 9/12
469/469 [==============================] - 42s 90ms/step - loss: 2.2038 - accuracy: 0.2098 - val_loss: 2.1427 - val_accuracy: 0.42
58
Epoch 10/12
469/469 [==============================] - 43s 91ms/step - loss: 2.1800 - accuracy: 0.2283 - val_loss: 2.1104 - val_accuracy: 0.46
21
Epoch 11/12
469/469 [==============================] - 45s 95ms/step - loss: 2.1537 - accuracy: 0.2468 - val_loss: 2.0738 - val_accuracy: 0.48
36
Epoch 12/12
469/469 [==============================] - 42s 89ms/step - loss: 2.1226 - accuracy: 0.2634 - val_loss: 2.0332 - val_accuracy: 0.50
44
Test loss: 2.0331578254699707
Test accuracy: 0.5044000148773193
new_train shape: (60000, 28, 28, 1)
new_test shape: (10000, 28, 28, 1)
Epoch 1/12
469/469 [==============================] - 48s 96ms/step - loss: 2.4078 - accuracy: 0.1008 - val_loss: 2.3010 - val_accuracy: 0.11
86
Epoch 2/12
469/469 [==============================] - 44s 94ms/step - loss: 2.3337 - accuracy: 0.1058 - val_loss: 2.2960 - val_accuracy: 0.12
43
Epoch 3/12
469/469 [==============================] - 44s 94ms/step - loss: 2.3116 - accuracy: 0.1146 - val_loss: 2.2947 - val_accuracy: 0.13
31
Epoch 4/12
```

```
469/469 [==============================] - 45s 96ms/step - loss: 2.3034 - accuracy: 0.1145 - val_loss: 2.2937 - val_accuracy: 0.13
88
Epoch 5/12
469/469 [==============================] - 42s 89ms/step - loss: 2.3019 - accuracy: 0.1160 - val_loss: 2.2934 - val_accuracy: 0.14
42
Epoch 6/12
469/469 [==============================] - 43s 91ms/step - loss: 2.2966 - accuracy: 0.1209 - val_loss: 2.2908 - val_accuracy: 0.15
30
Epoch 7/12
469/469 [==============================] - 44s 94ms/step - loss: 2.2956 - accuracy: 0.1215 - val_loss: 2.2885 - val_accuracy: 0.16
19
Epoch 8/12
469/469 [==============================] - 41s 87ms/step - loss: 2.2932 - accuracy: 0.1249 - val_loss: 2.2849 - val_accuracy: 0.17
24
Epoch 9/12
469/469 [==============================] - 43s 92ms/step - loss: 2.2913 - accuracy: 0.1280 - val_loss: 2.2813 - val_accuracy: 0.18
08
Epoch 10/12
469/469 [==============================] - 44s 94ms/step - loss: 2.2877 - accuracy: 0.1319 - val_loss: 2.2766 - val_accuracy: 0.18
82
Epoch 11/12
469/469 [==============================] - 40s 86ms/step - loss: 2.2835 - accuracy: 0.1389 - val_loss: 2.2707 - val_accuracy: 0.19
91
Epoch 12/12
469/469 [==============================] - 44s 93ms/step - loss: 2.2785 - accuracy: 0.1432 - val_loss: 2.2632 - val_accuracy: 0.21
24
Test loss: 2.2632203102111816
Test accuracy: 0.21240000426769257
new_train shape: (60000, 28, 28, 1)
new_test shape: (10000, 28, 28, 1)
Epoch 1/12
469/469 [==============================] - 46s 91ms/step - loss: 2.5830 - accuracy: 0.0997 - val_loss: 2.3157 - val_accuracy: 0.10
33
Epoch 2/12
469/469 [==============================] - 41s 87ms/step - loss: 2.3737 - accuracy: 0.1032 - val_loss: 2.3021 - val_accuracy: 0.10
15
Epoch 3/12
469/469 [==============================] - 43s 93ms/step - loss: 2.3282 - accuracy: 0.1044 - val_loss: 2.3014 - val_accuracy: 0.10
45
Epoch 4/12
469/469 [==============================] - 42s 89ms/step - loss: 2.3144 - accuracy: 0.1040 - val_loss: 2.3019 - val_accuracy: 0.10
30
Epoch 5/12
469/469 [==============================] - 41s 88ms/step - loss: 2.3095 - accuracy: 0.1021 - val_loss: 2.3021 - val_accuracy: 0.10
69
Epoch 6/12
469/469 [==============================] - 43s 92ms/step - loss: 2.3066 - accuracy: 0.1050 - val_loss: 2.3023 - val_accuracy: 0.11
00
Epoch 7/12
469/469 [==============================] - 41s 87ms/step - loss: 2.3058 - accuracy: 0.1042 - val_loss: 2.3023 - val_accuracy: 0.11
17
Epoch 8/12
469/469 [==============================] - 42s 90ms/step - loss: 2.3051 - accuracy: 0.1068 - val_loss: 2.3024 - val_accuracy: 0.11
27
Epoch 9/12
469/469 [==============================] - 44s 93ms/step - loss: 2.3044 - accuracy: 0.1079 - val_loss: 2.3024 - val_accuracy: 0.10
97
Epoch 10/12
469/469 [==============================] - 40s 86ms/step - loss: 2.3042 - accuracy: 0.1059 - val_loss: 2.3024 - val_accuracy: 0.11
20
Epoch 11/12
469/469 [==============================] - 45s 97ms/step - loss: 2.3035 - accuracy: 0.1064 - val_loss: 2.3024 - val_accuracy: 0.11
17
Epoch 12/12
469/469 [==============================] - 44s 93ms/step - loss: 2.3032 - accuracy: 0.1078 - val_loss: 2.3024 - val_accuracy: 0.11
17
Test loss: 2.302367687225342
Test accuracy: 0.11169999837875366
```

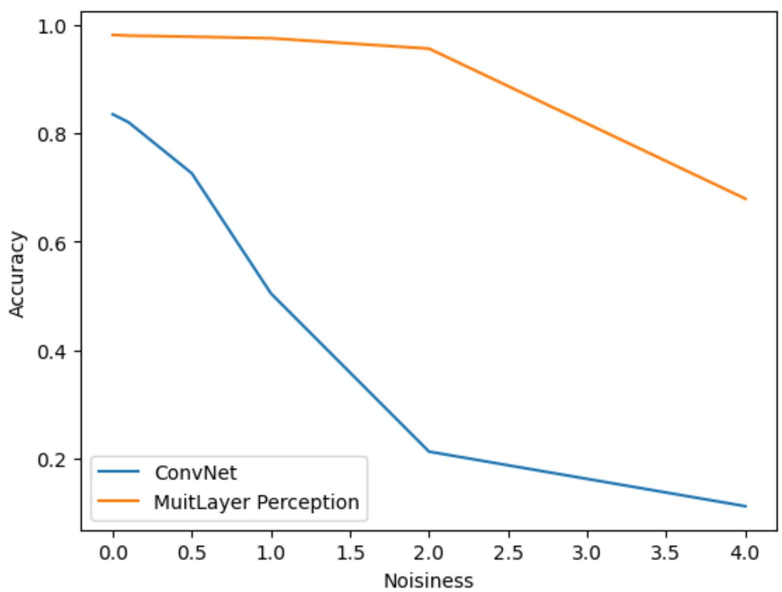
In [110…
```
losses_MLP = [0.0719, 0.1039, 0.1362, 0.1539, 0.2294, 0.8322]
acc_MLP = [0.9815, 0.9801, 0.9781, 0.9752, 0.9561, 0.6791]
```

In [111…
```
plt.figure()
plt.plot(scale, losses, label="ConvNet")
plt.plot(scale, losses_MLP, label="MuitLayer Perception")
plt.xlabel('Noisiness')
plt.ylabel('Losses')
plt.legend()
plt.show()
```



In [112…

```
plt.figure()
plt.plot(scale, acc, label="ConvNet")
plt.plot(scale, acc_MLP, label="MuitLayer Perception")
plt.xlabel('Noisiness')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



The number of epochs used for this assignment (ConvNet) is smaller than last week's assignment using MultiLayer Perception - 12 epochs for ConvNet compared to 20 for MultiLayer Perception, but even so the accuracy is much higher and the losses lower for MultiLayer Percenption within 12 epochs as noise increases. With no noise at all, the MultiLayer Perception at 12 epochs had 98% accuracy and less than 0.1 in losses where the ConvNet with no noise had an accuracy at 83% and 0.7 in losses.