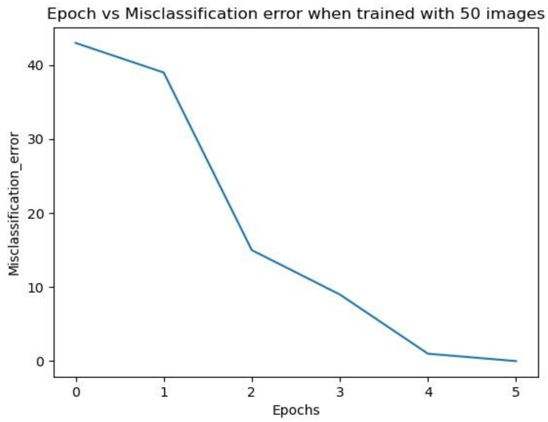
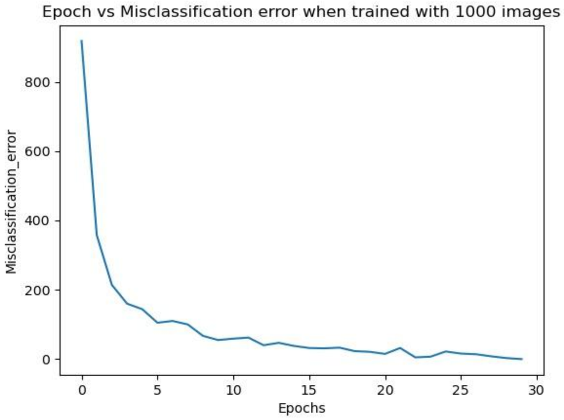
f) n = 50, lr = 1, e = 0

Misclassification error on test dataset when trained with 50 images is 4428 which is 44.28% which is very high. It is high because the network was trained on fewer examples and also the training error converged to zero and the test misclassification error is close to 45% this means that the network overfit the training data hence it did not generalize to the test data hence the high loss on test data.



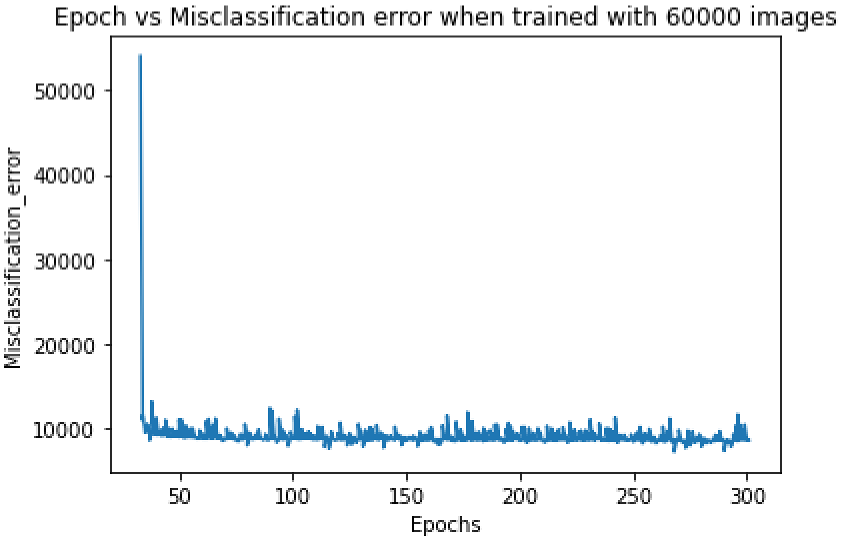
g) n = 1000, lr = 1, e = 0

Misclassification error on test dataset when trained with 1000 images is 1789 which is 17.89%. As we can see the test error decreased from 50 examples as we have trained the network with more data it was able to generalize better than 50 examples. And we can also see that it took more epochs to converge compared to 5 examples.



h) n = 60000, lr = 1, e = 0

When using all the training examples the algorithm will not converge and one of the reasons is that there is no nonlinearity introduced in the network and we are not treating image as a two dimensional array rather we are flattening it by doing this we are losing spatial information in an image. By flattening it, the relation between adjacent pixels is not captured hence the algorithm is not converging.



i) n = 60000, lr = 1, e = 0.15

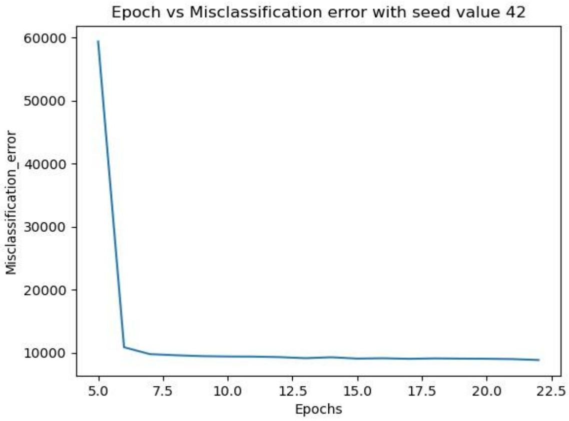
Seed values used 42,84,168 to give different initial weight values.

Seed 42:

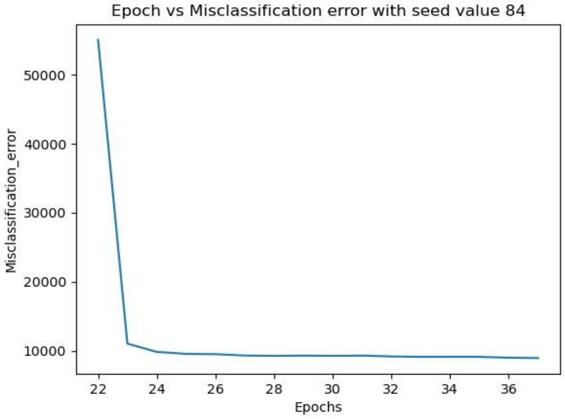
Epoch at which the algorithm converged 22

Errors at the time of convergence 8872

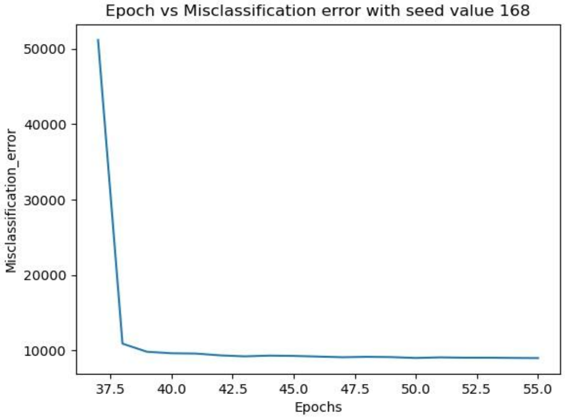
Misclassification error on test dataset with seed value 42 1714



Seed 84:
Epoch at which the algorithm converged 37
Errors at the time of convergence 8953
Misclassification error on test dataset with seed value 84 1839



Seed 168:
Epoch at which the algorithm converged 55
Errors at the time of convergence 8987
Misclassification error on test dataset with seed value 168 1479



As we can see from the graphs as the initial weight changed the epoch at which the algorithm converged changed and the misclassification error on the test dataset also changed.

Code:

```
import gzip
import numpy as np
import os
import matplotlib.pyplot as plt


# code taken from
https://stackoverflow.com/questions/40427435/extract-images-from-idx3-ubyte-file-or-gzip-via-python
def read_images(train=True):
    if train:
        path = 'dataset/train-images-idx3-ubyte.gz'
    else:
        path = 'dataset/t10k-images-idx3-ubyte.gz'
    with gzip.open(os.path.join(os.path.dirname(os.getcwd()), path), 'r') as f:
        # first 4 bytes is a magic number
        magic_number = int.from_bytes(f.read(4), 'big')
        # second 4 bytes is the number of images
        image_count = int.from_bytes(f.read(4), 'big')
        # third 4 bytes is the row count
        row_count = int.from_bytes(f.read(4), 'big')
        # fourth 4 bytes is the column count
        column_count = int.from_bytes(f.read(4), 'big')
        # rest is the image pixel data, each pixel is stored as an unsigned byte
        # pixel values are 0 to 255
        image_data = f.read()
        images_array = np.frombuffer(image_data, dtype=np.uint8) \
            .reshape((image_count, row_count, column_count))
        return images_array


def read_labels(train=True):
    if train:
        path = 'dataset/train-labels-idx1-ubyte.gz'
    else:
        path = 'dataset/t10k-labels-idx1-ubyte.gz'
    with gzip.open(os.path.join(os.path.dirname(os.getcwd()), path), 'r') as f:
        # first 4 bytes is a magic number
        magic_number = int.from_bytes(f.read(4), 'big')
        # second 4 bytes is the number of labels
        label_count = int.from_bytes(f.read(4), 'big')
        # rest is the label data, each label is stored as unsigned byte
        # label values are 0 to 9
        label_data = f.read()
        labels_array = np.frombuffer(label_data, dtype=np.uint8)
        return labels_array


images, labels = read_images(), read_labels()
test_images, test_labels = read_images(train=False), read_labels(train=False)

n = [60000, 1000, 50]

lr = 1
seed = 42
np.random.seed(seed)
for no_of_examples in [60000, 1000, 50]:
    misclassification_per_epoch = []
    epochs = []
    epoch, errors = 0, 0
    W = np.random.uniform(low=-1, high=1, size=(10, 784))
    if no_of_examples != 60000:
        e = 0
    else:
        e = 0.0
    training_images, training_labels = images[:no_of_examples], labels[:no_of_examples]
    for image, label in zip(training_images, training_labels):
        image = image.reshape(784, 1)
        v = np.matmul(W, image)
        predicted_label = np.argmax(v)
        if predicted_label != label:
            errors += 1
    misclassification_per_epoch.append(errors)
    epochs.append(epoch)
    while True:
        errors = 0
        epoch += 1
        for image, label in zip(training_images, training_labels):
            image = image.reshape(784, 1)
            v = np.matmul(W, image)
            predicted_label = np.argmax(v)
            if predicted_label != label:
                errors += 1
                d = np.zeros((10, 1))
                d[label] = 1
                u_wx = np.heaviside(v, 0).reshape(10, 1)
                W = W + lr * np.matmul((d - u_wx), np.transpose(image))
        misclassification_per_epoch.append(errors)
        epochs.append(epoch)
        if errors / no_of_examples <= e:
            print("Epoch at which the algorithm converged", epoch)
            print("Misclassification error at the time of convergence when trained with  " +
str(no_of_examples) + " images", errors)
            break
    plt.plot(epochs, misclassification_per_epoch)
    plt.xlabel("Epochs")
    plt.ylabel("Misclassification_error")
    plt.title("Epoch vs Misclassification error when trained with " + str(no_of_examples) + "
images")
    plt.savefig("mce_epochs_{}.jpg".format(no_of_examples))
    plt.show()
    errors = 0
    for image, label in zip(test_images, test_labels):
        image = image.reshape(784, 1)
        v = np.matmul(W, image)
        predicted_label = np.argmax(v)
        if predicted_label != label:
            errors += 1
    print("Misclassification error on test dataset when trained with " + str(no_of_examples) + "
images", errors)

e = 0.15

for seed in [42, 84, 168]:
    misclassification_per_epoch = []
    epochs = []
    np.random.seed(seed)
    W = np.random.uniform(low=-1, high=1, size=(10, 784))
    training_images, training_labels = images, labels
    for image, label in zip(training_images, training_labels):
        image = image.reshape(784, 1)
        v = np.matmul(W, image)
        predicted_label = np.argmax(v)
        if predicted_label != label:
            errors += 1
    misclassification_per_epoch.append(errors)
    epochs.append(epoch)
    while True:
        errors = 0
        epoch += 1
        for image, label in zip(training_images, training_labels):
            image = image.reshape(784, 1)
            v = np.matmul(W, image)
            predicted_label = np.argmax(v)
            if predicted_label != label:
                errors += 1
                d = np.zeros((10, 1))
                d[label] = 1
                u_wx = np.heaviside(v, 0).reshape(10, 1)
                W = W + lr*np.matmul((d-u_wx), np.transpose(image))
        misclassification_per_epoch.append(errors)
        epochs.append(epoch)
        if errors/60000 <= e:
            print("Epoch at which the algorithm converged", epoch)
            print("Errors at the time of convergence", errors)
            break
    plt.plot(epochs, misclassification_per_epoch)
    plt.xlabel("Epochs")
    plt.ylabel("Misclassification_error")
    plt.title("Epoch vs Misclassification error with seed value " + str(seed))
    plt.savefig("mce_epochs_seed_{}.jpg".format(seed))
    plt.show()
    errors = 0
    for image, label in zip(test_images, test_labels):
        image = image.reshape(784, 1)
        v = np.matmul(W, image)
        predicted_label = np.argmax(v)
        if predicted_label != label:
            errors += 1
    print("Misclassification error on test dataset with seed value " + str(seed), errors)
```