Nitzan Gorani 206020489, Sivan Geva 313581589

336546- Machine Learning in Health Care- HW4

Theoretical questions

Part 1: Fully connected layers

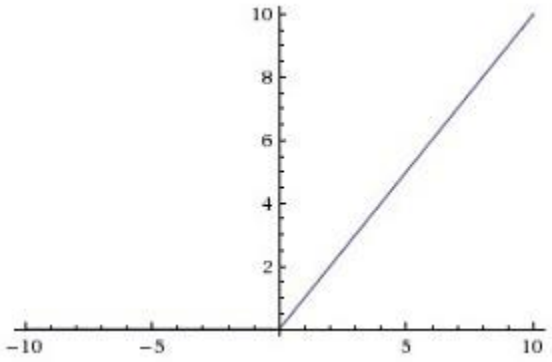
Task 1: NN with fully connected layers

```
Test Loss of ReLU+batch 64 after training is 0.79
Test Accuracy of ReLU+batch 64 after training is 66.29 %
```

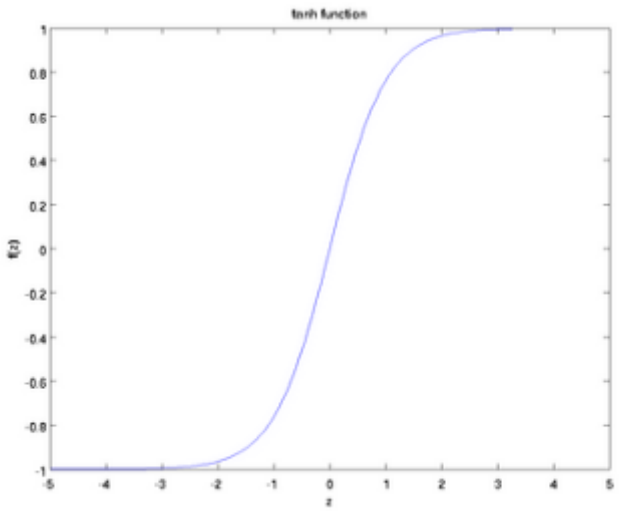
Task 2: Activation functions

We chose 'tanh' activation function.

ReLU activation function looks like this:



tanh activation function look like this:



We see that for negative values of X, ReLU turns 0 activation, which means fewer neurons are firing (sparse activation). For tanh activation function, there's a range were the activation function is steep and towards either end of the function, the gradient gets smaller. Thus, at the edges the network doesn't learn much more or learns extremely slow (namely vanishing gradient problem).

Pros of tanh:

- Nonlinear- gradient isn't constant (good for multilayer NN)
- Bounded [-1,1]

Cons of tanh:

- Vanishing gradient problem.
- Dense activation- costly.

Pros of ReLU:

- Nonlinear- although for positive values of X derivative is constant.
- Sparse activation and efficient- less computational operations.

Cons of ReLU:

- Not bound- can blow up the activation. $[0, \infty)$
- "Dying ReLu problem"- several neurons die and not respond.

```
Test Loss of tanh+epochs=25 after training is 0.83
Test Accuracy of tanh+epochs=25 after training is 65.71 %
```

Task 3: Number of epochs

For using 25 epochs the performance was: (see above)

For using 40 epochs the performance was:

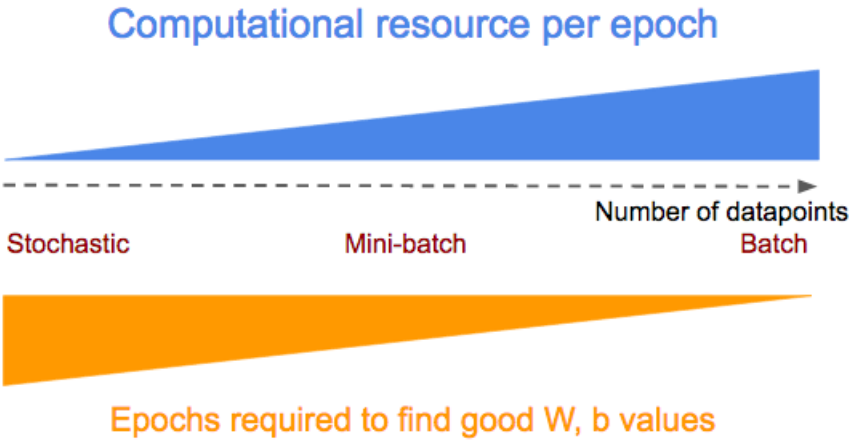
```
Test Loss of tanh+epochs=40 after training is 0.83
Test Accuracy of tanh+epochs=40 after training is 64.57 %
```

We see that the performance was slightly better using 25 epochs then using 40 epochs. More epochs mean we train the model more times on the same training set. Maybe for this net architecture and for this specific data set, training the model for 40 epochs doesn't improve performance on the testing set due to overfitting.

Task 4: Mini-batches

Reducing batch size from 64 to 32 makes smaller gradient steps for the same number of samples seen. The results in this case showed no improvement in performance.

Stochastic gradient decent is mini batch that consist only one example before updating the weights according to the updated gradient. Mini batch consist a group of examples (more than one) that we calculate the cumulative loss and add to the gradient (linear operation). The advantage of using mini-batch vs. SGD is that mini-batch is less noisy and less affected by changes. Unfortunately, it requires higher computational resources and takes longer time. As a tradeoff, usually mini batch has small amount of examples. In addition, SGD has faster convergence but usually it converges to unstable local minimum and not global minimum.




```
Test Loss of ReLU+batch=32 after training is 0.86
Test Accuracy of ReLU+batch=32 after training is 64.57 %
```

Task 5: Batch normalization

Batch normalization applies standardization on the activation of each layer allowing to learn from more stable distribution of inputs, accelerating the learning process and the outcomes. In addition, batch normalization provides a good way to reduce internal covariance shift. We noticed that the number of epochs was different from tanh activation in the previous sections, we think it had influenced the result as well.

```
Test Loss of tanh+Batch Normalization after training is 0.91
Test Accuracy of tanh+Batch Normalization after training is 66.29 %
```

Part 2: Convolutional Neural Network (CNN)

Task 1: 2D CNN

a. The model has 8 layers: 5 CNN, 3 fully connected NN.
b. The number of filters in each layer is:
   Conv2D_1 has 64 filters, Conv2D_2 has 128 filters, Conv2D_3 has 128 filters, Conv2D_4 has 256 filters, Conv2D_5 has 256 filters.
c. The number of parameters won't be similar between 2D CNN to fully connected NN. 2D CNN uses relatively fewer parameters to train[1].

$$\#parameters\ of\ 2D\ CNN = (filter\_width * filter\_height * num.\_filters\_previous\_layer + \underbrace{1}_{bias}) * num.\_filters\_current\_layer$$

[1]https://towardsdatascience.com/understanding-and-calculating-the-number-of-parameters-in-convolution-neural-networks-cnns-fc88790d530d

$$\#parameters\ of\ fully\ connected\ NN = (current\_layer\_neurons + \underbrace{1}_{bias}) * previous\_layer\_neurons$$

d. This specific NN is performing regularization using two methods:
   1. Dropout- ignores neurons during the training phase of certain set of neurons which is chosen at random. In this case, dropRate=0.3 which means 30% of the neurons in the layer are ignored.
   2. 'L2' - tries to reduce the possibility of overfitting by keeping the values of the weights small.

Task 2: Number of filters

Performance of the original filters:

```
2D CNN: test loss, test acc: [8.120816230773926, 0.2914285659790039]
```

Performance after filter reduction:

```
2D CNN (filter reduce)- test loss, test acc: [4.765956878662109, 0.38857144117355347]
```

Reducing the number of filters reduced the model's complexity and improved performance (see results above). We assume that the model was overfitted before the filter reduction, thus lowering the number of filters favored the model's generalization.
In addition we notice that in some running the ACC went down therefore our assumption is inconclusive, maybe because the quality of the data is not good.