1. Introduction

The aim of the laboratory was to familiarize with designing Neural Network with linear regression using TensorFlow.

2. Results

The code was based on an online tutorial, therefore most of it was omitted.

Tutorial required to make some linear data, split it and train NN to calculate regression. To add more to it, firstly noise was added to data.

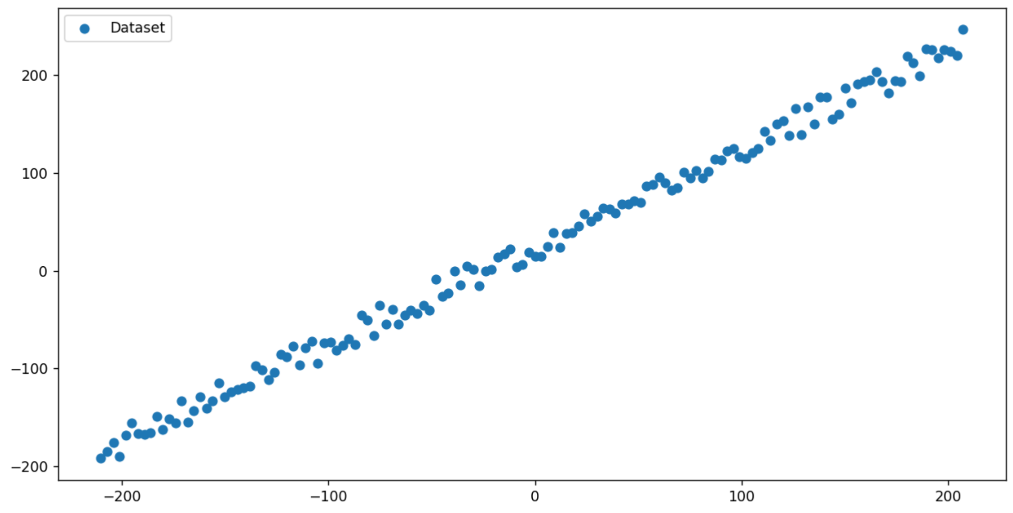

Figure 1 Data with noise

Then, signal was split into test and train parts in 110:30 ratio.

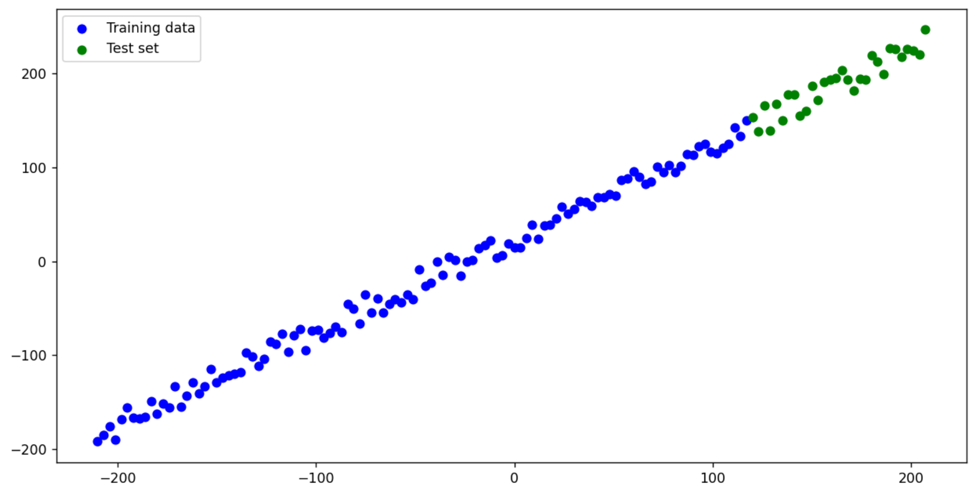

Figure 2 Split dataset

Next, simple NN was created:

```
model = tf.keras.Sequential([tf.keras.layers.InputLayer(
    input_shape=1),
    tf.keras.layers.Dense(1)])
```

After that, there was need for experimenting. We have to find optimal number of epochs and learning rate. Also choice of optimizer is important as well.

Number of epochs was chosen to be 300 as this value did not cause overfitting. Then, 2 optimizers were tested: SGD (Stochastic Gradient Descent) and Adam. For both of them, different learning rates were checked.

```
number_of_epochs = 300
History = []
r2_scores = []
learning_rates = [0.2, 0.1, 0.01, 0.001]
for learning in learning_rates:
    #compiling
    model.compile( loss = tf.keras.losses.mae,
                optimizer = tf.keras.optimizers.SGD(learning_rate=learning),#SGD-
> stochastic gradient descent
                #optimizer = tf.keras.optimizers.Adam(learning_rate=learning),
                metrics = ['mae'])

    # tensorflow run model/train model on input data
    history = model.fit(tf.expand_dims(X_train, axis=-1), y_train,
epochs=number_of_epochs)
    History.append(history)

    # Prediction of neural network model
    preds = model.predict(X_test)
    r2 = r2_score(y_test, preds)
    print('R score is :', r2)
    r2_scores.append(r2)
```

Below, Loss parameter can be seen for SGD optimizer:

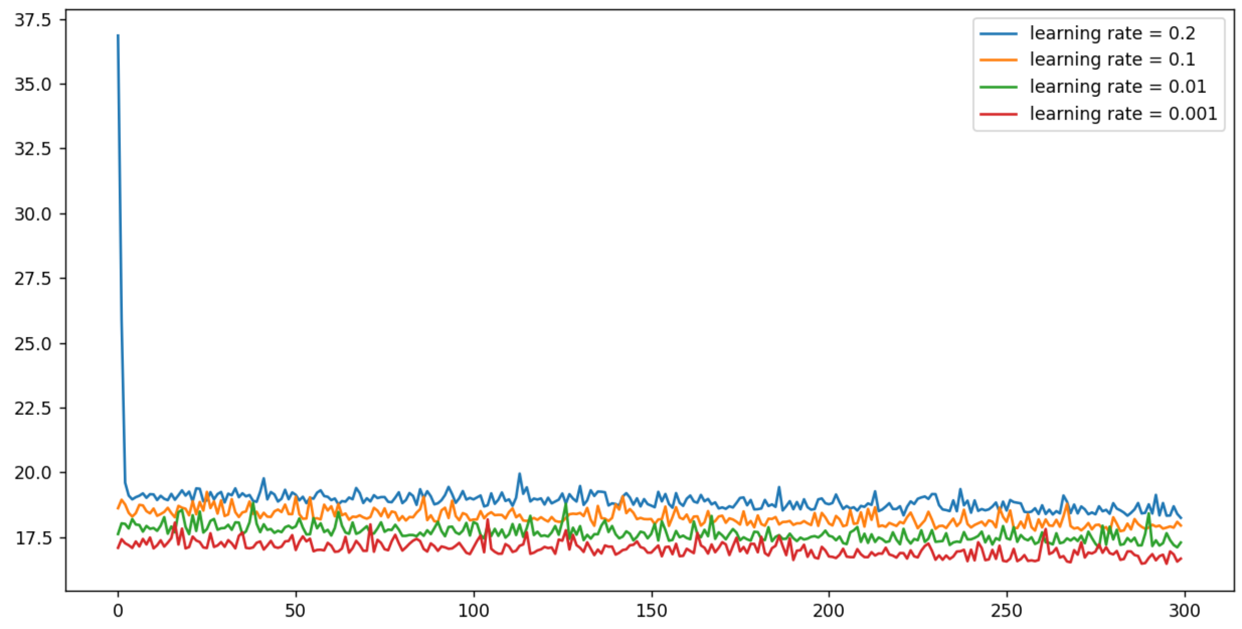

Figure 3 Loss vs Learning rate for SGD

Also R-square score was calculated:

Learning rate	0.2	0.1	0.01	0.001
R-square	-7594.382	-805.543	-6.205	0.688

Both metrics tells us that in case of SGD, the lower learning rate, the better performance (up to some point)

Alternatively the same was done for Adam optimizer:

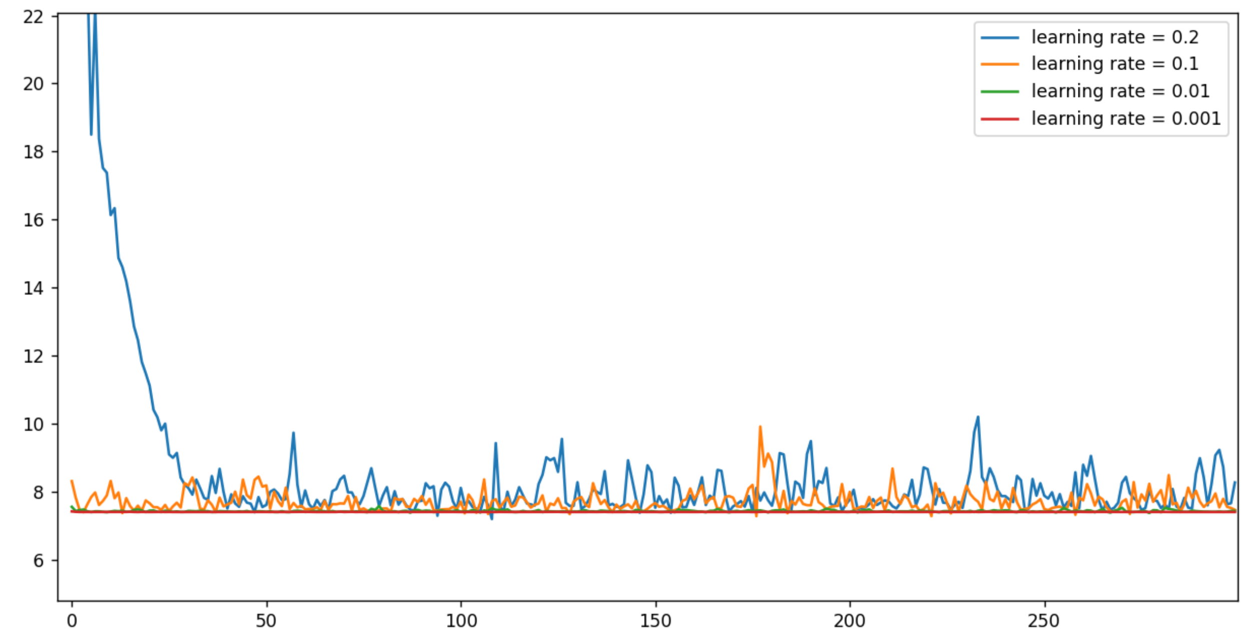

Figure 4 Loss vs Learning rate for Adam (zoomed in)

Learning rate	0.2	0.1	0.01	0.001
R-square	0.879	0.905	0.881	0.890

As shown, Adam optimizer performed better. Contrary to SGD, Adam performed better with higher learning rate.

Using the best parameters, models for both optimizer were used to predict testing part of dataset.

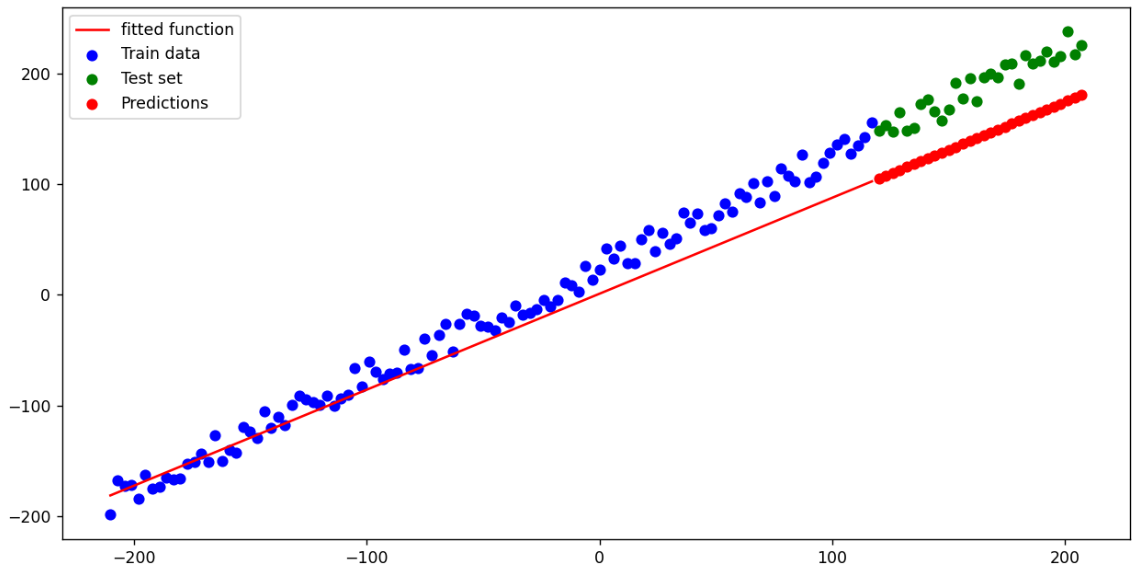

Figure 5 Prediction for SGD

Function for SGD:

$$y = 0.867586 * x + 0.867542$$

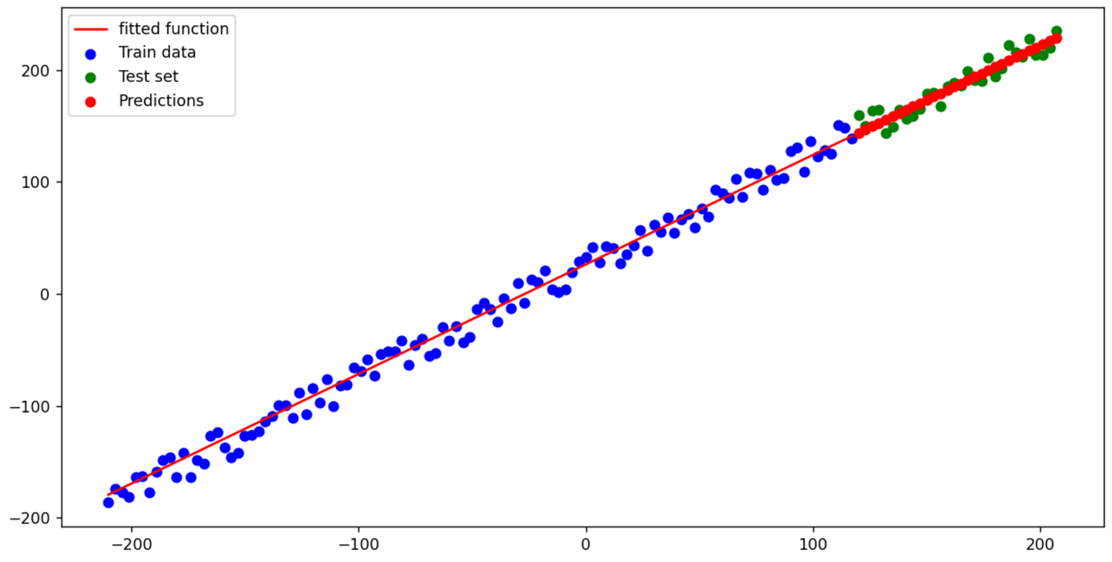

Figure 6 Prediction for Adam

Function for Adam:

$$y = 0.979848 * x + 26.329998$$

As one can see, Adam optimizer predicted trend with better accuracy.