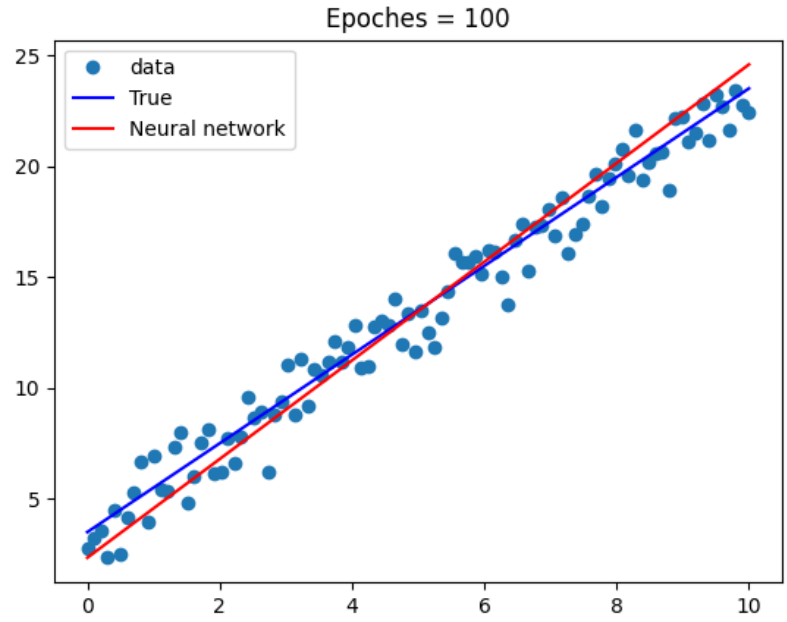
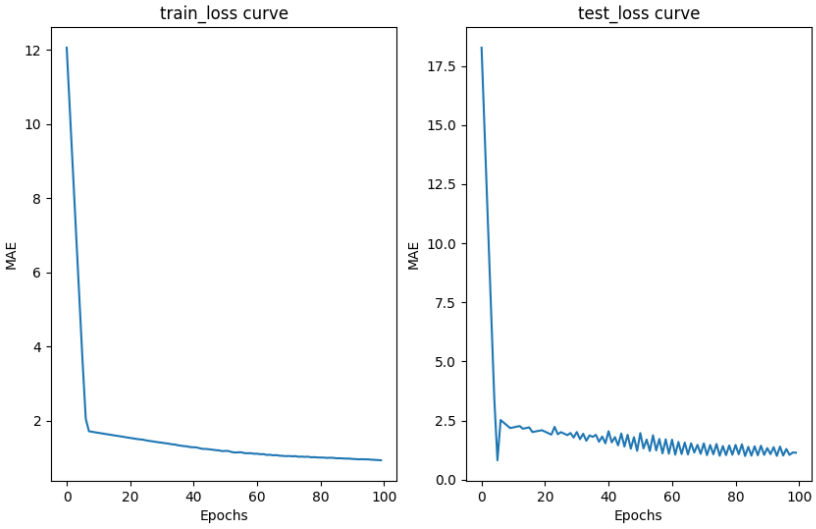
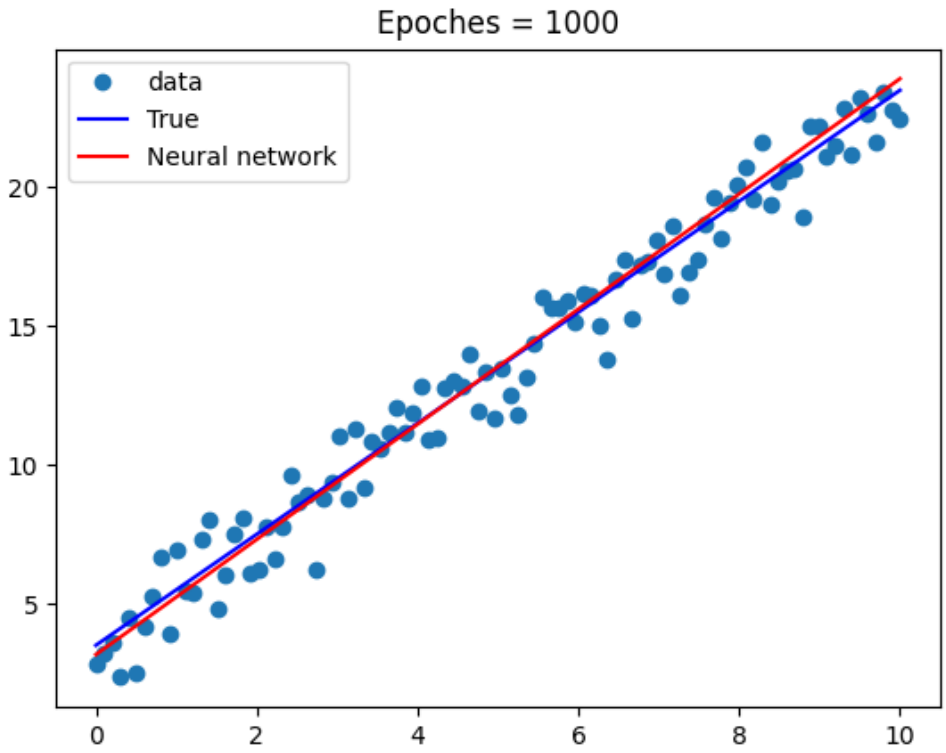
Report on the completion of the third task.

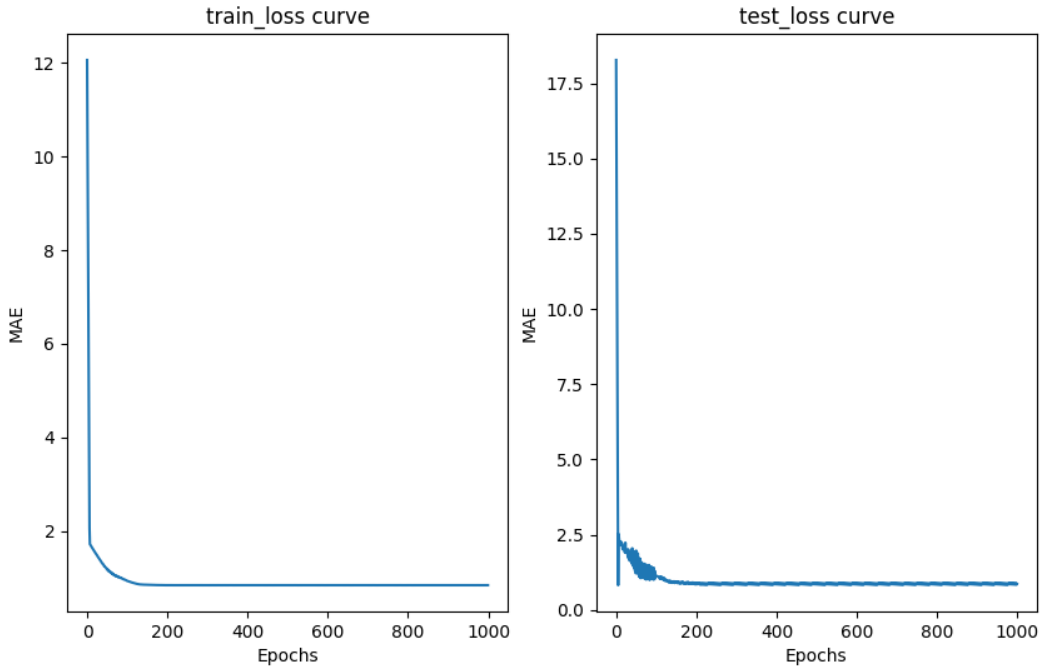
1. **The purpose of the task.**
   The main goal of the task is to create train and test loss curves, plot the neural network performance on the data and investigate how does the absence of training one parameter affect the other.
2. **Plotting the graphs of model performance.**






```
Original coefficients:
        weights = 2, bias = 3.5
Model's start coefficients:
        weights = 0.4390980005264282, bias = -0.8004615306854248
Model's final coefficients:
        weights = 2.2234399318695 07, bias = 2.344538688659668
```

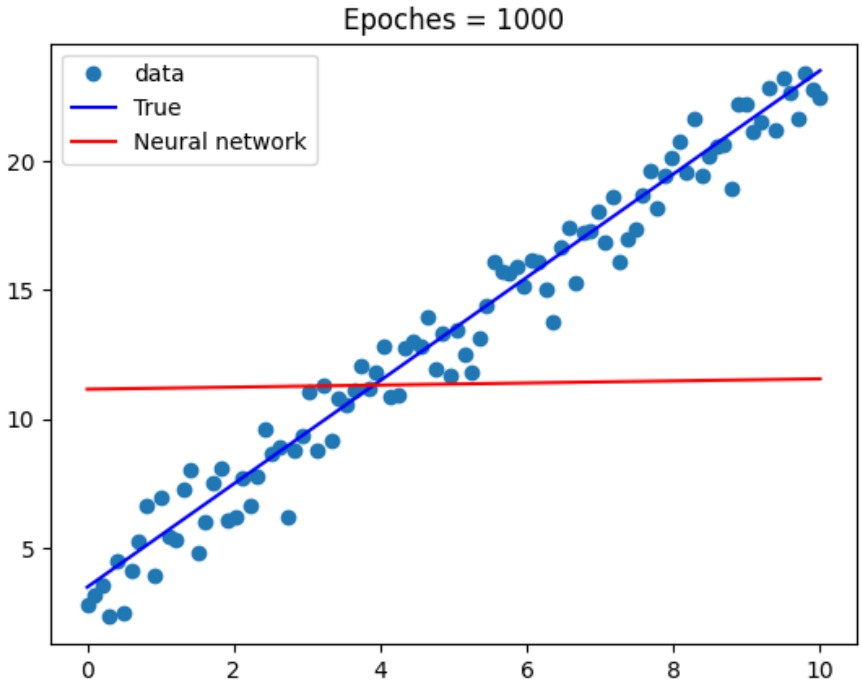


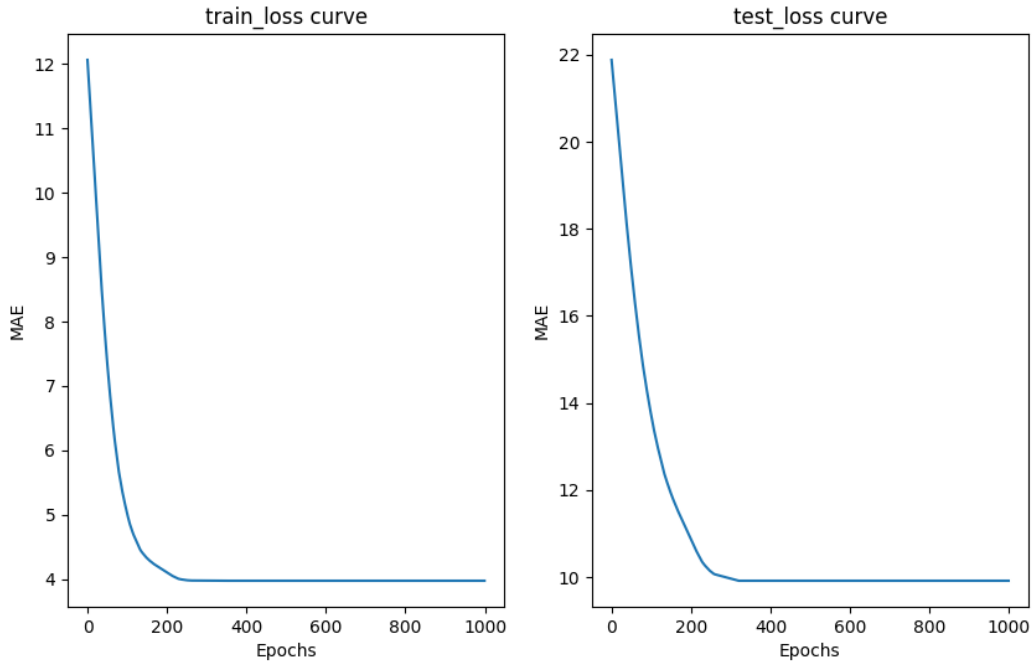



```
Original coefficients:
        weights = 2, bias = 3.5
Model's start coefficients:
        weights = 0.4390980005264282, bias = -0.8004615306854248
Model's final coefficients:
        weights = 2.074979543685913, bias = 3.1645431518554688
```

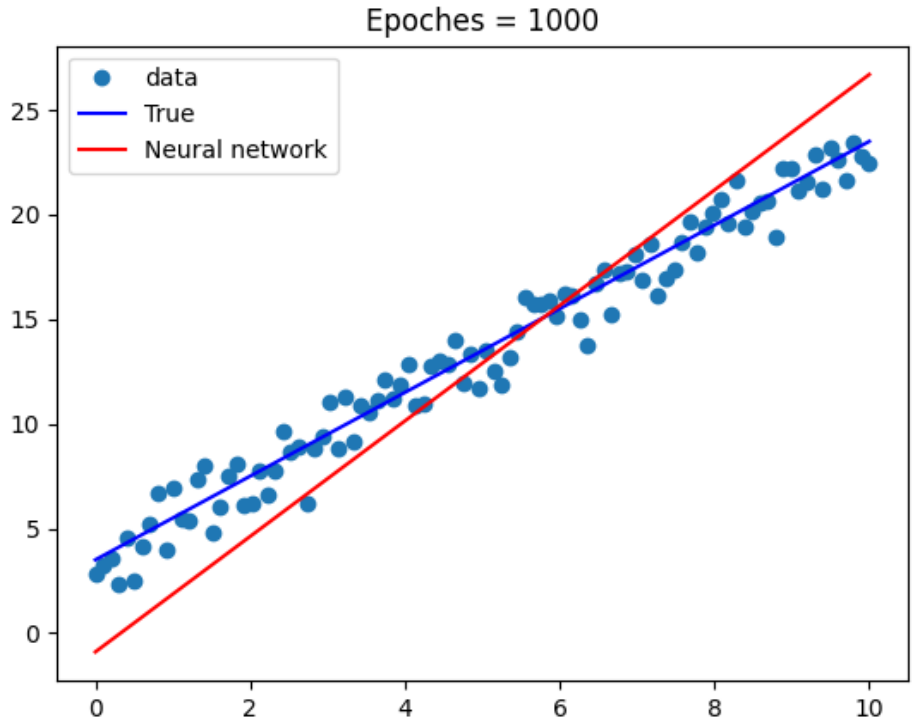
3. Let's try not to train some parameter.

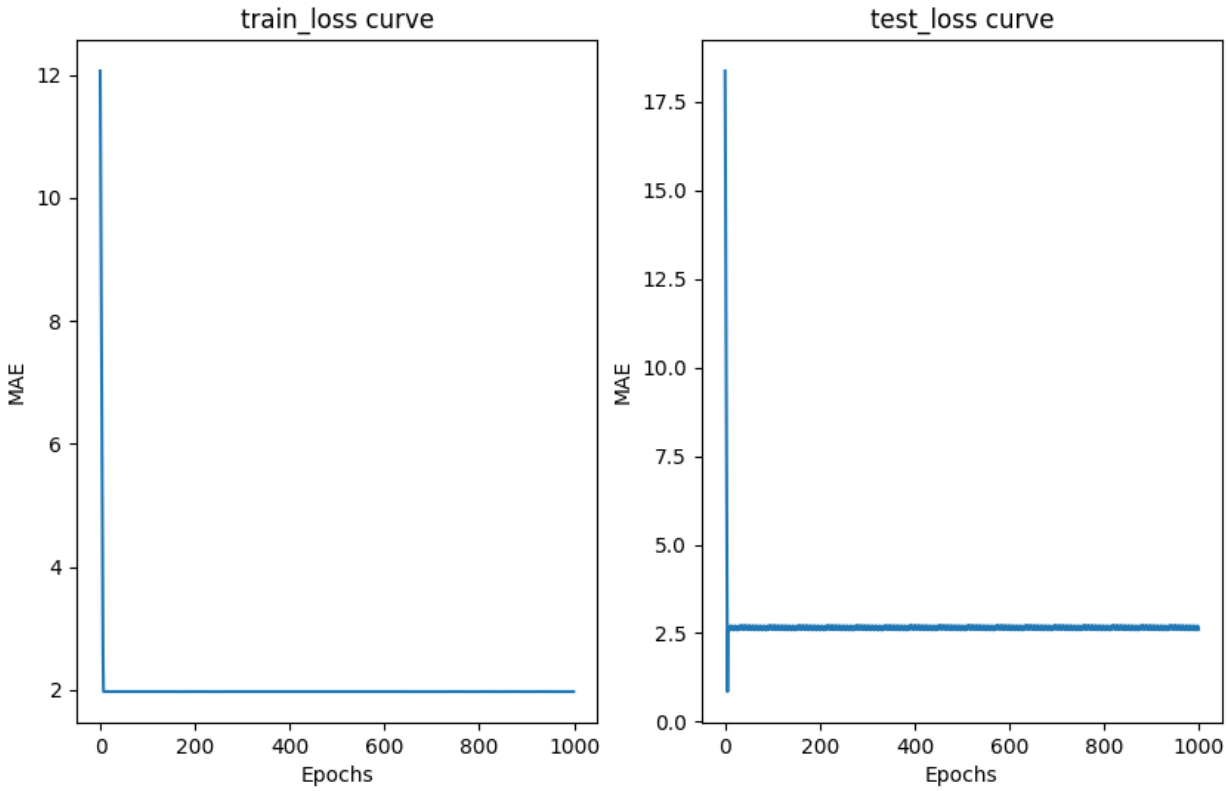
Model without training a slope





Model without training a bias





4. **Conclusion.**

- From the graphs above, we can see that the train loss curve is actually softer than the test one on the smaller number of epochs, but by increasing them, these plots

start to have the same softness and the red line of the neural network fits the data slightly better.
-If we stop to train weights, the slope between the line and the x axis becomes 0 and the model tries to fit the data just by the intercept. This causes big losses, even though for 1000 epochs, loss curves dropped heavily.
-On the other hand, training weights without bias gives us the opposite – the model tries to fit the data just by finding the optimal slope. This also causes a pretty bad line and the loss of around 2 on train and around 2.5 on test data, but total loss of the model is much lower than when training only intercept.