**Answer in your Report:** Why should we add the momentum to SGD? What is the main two improvments adam has comparing to the simple SGD? What was the best optimization algorithm and best learning rate you've found?

**Adding momentum to SGD makes it possible to avoid resetting the gradient when finding a local minimum point, thus making it possible to find the global minimum point.**

**The improvements one has on SGD:**

1. **Adding a factor to normalize the "zig-zag" problem and thus move quickly towards the global minimum point and deviate less from the direct direction to the global minimum point according to the gradient of each batch**
2. **Normalize the normalizing factor so that it does not increase continuously**

**Best optimization algorithm - ADAM, best learning rate - 0.001.**

**Answer in your Report:** What is the purpose of batch normalization (why do we use it)? Did the Batch normalization improve the network preformance? Summeraize this section results in your report.

**The purpose of normalization is to contribute to convergence, enable higher learning rates, reduce the initialization effect on the weights process and function as regularization (reducing overfit). In this case, since there is no significant overfit, it can be seen that the normalization did not significantly improve the performance of the network.**

**Answer in your Report:** Why should we use regularization? How does the regularization affect the train accuracy and loss? How does it affect the val and test accuracy and loss? What was the best regulrization method? Summarize this section results in your report.

Adding a regularization factor helps us choose the simpler solutions according to our preference. Adding the regularization caused the accuracy values to increase while the accuracy values decreased. In this experiment it seems that the two regularization methods had the same effect on the accuracy and loss values and therefore in this case both are equally good.

Summary Up To Here

**Answer in your Report:** What is your best architecture? what its best accuracy and loss on the test set? Compare your result to assignment 2. How did you mange to improve the model? Summarize this section results in your report.

The best architecture was the one created after adding the regularization. Its characteristics:

Accuracy: 74.27, Loss: 0.7656377398567323

One of the reasons the network in Exercise 2 failed to learn was that it lacked regularization. In this exercise, the model was improved by adding two types of regulation - Batch Norm and Dropout\L2. Following this, the network was able to perform learning that was reflected in the increase in its accuracy and the decrease in loss percentages.

**Answer in your Report:** What was the pretrained ResNet50 accuracy and loss? Is it overfit/underfit/well-fit the data? Has it got better accuracy than the non-pretrained ResNet? Summarize this section results in your report.

Accuracy test: 80.15, Loss: 0.6651505689724857

Overfit is characterized by a low error in training and a high error in the test, and these characteristics can be seen in this experiment (the accuracy of the training was slightly higher than the accuracy of the test since ResNet50 is trained by 50 convolution layers), so it can be said that there is a minor overfit for the data.

Also, (and as can be seen) ResNet50 has better accuracy than the untrained ResNet.

## Project Summary

**Answer in your Report:** Write a short summerization of your attempts. What worked for you? which architecture and hyperparameters led to the best preformance?

The way that worked for me was to try to reach the best hyperparameters by means of coarse tuning until the desired range (with the minimum loss) is understood and then fine tuning until finding the lowest loss.

The best architecture was the one we used in RESNET50 with the hyperparameters: LR=0.0003 and bs\_train=512.

It led to the best performance: accuracy=80.15 and loss=0.6651505689724857.