ResNets

ResNets are fully convolutional neural networks introduced by Microsoft Research in 2015, these models outperformed almost all existing architectures and still considered the state of the art.

The idea is deep residual learning framework which consist of many residuals blocks of size two or three layers, each block has a shortcut connection between these layers like on the figure 2 from the original paper. The output of previous layer is added to the output of the last layer in the block and then ReLU activation is applied.

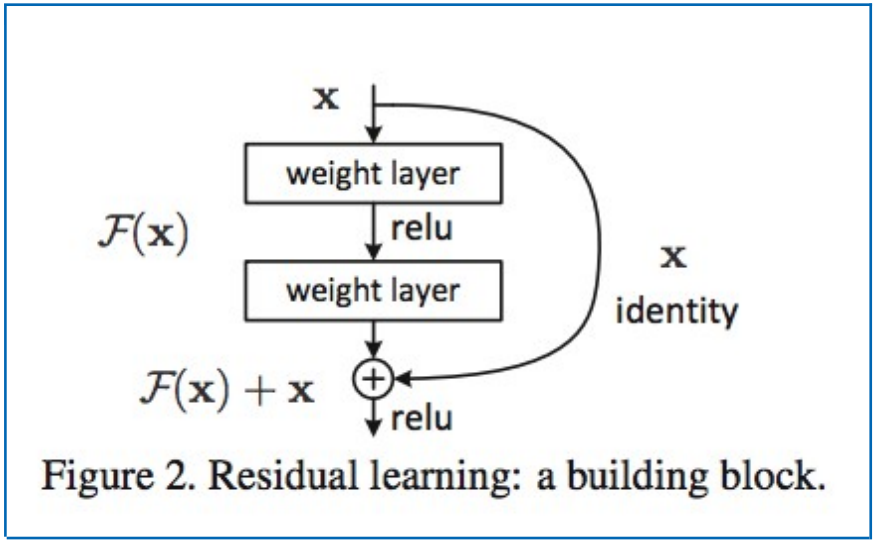
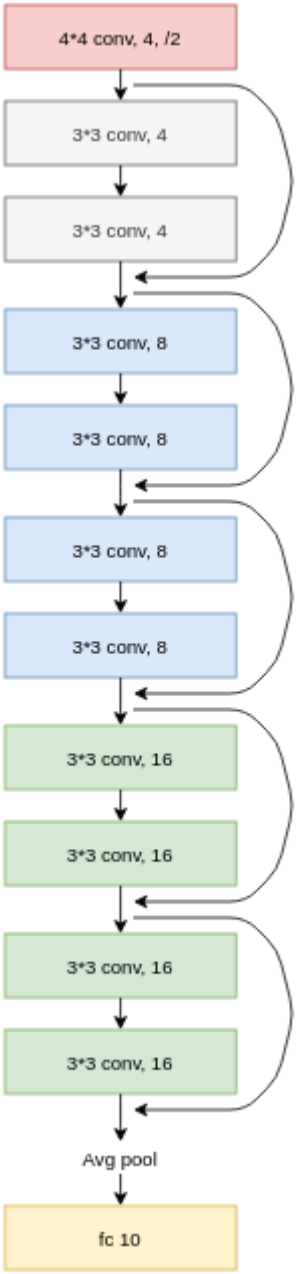


Figure 2. Residual learning: a building block.

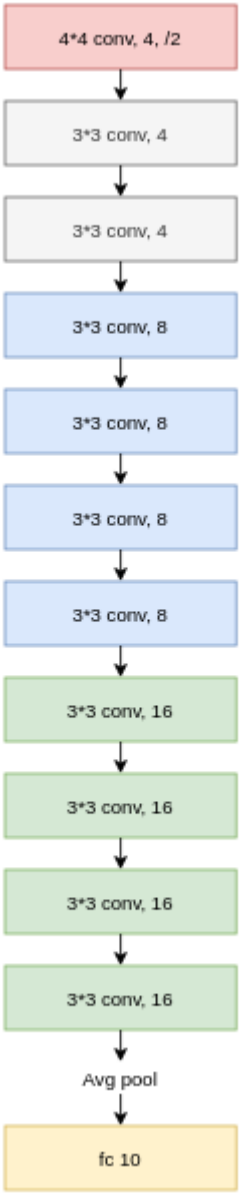
The researches considered from 18 to 152 layers networks and even tried a 1202 layers network, they showed that the error tends to decrease with increasing depth whereas it is not the case for plain fully convolutional networks.



I built a ResNet with 12 layers (5 residual blocks) since the performance is better comparing to the plain version of the network and it is feasible to compute smoothness of each layer with computational resources I have.

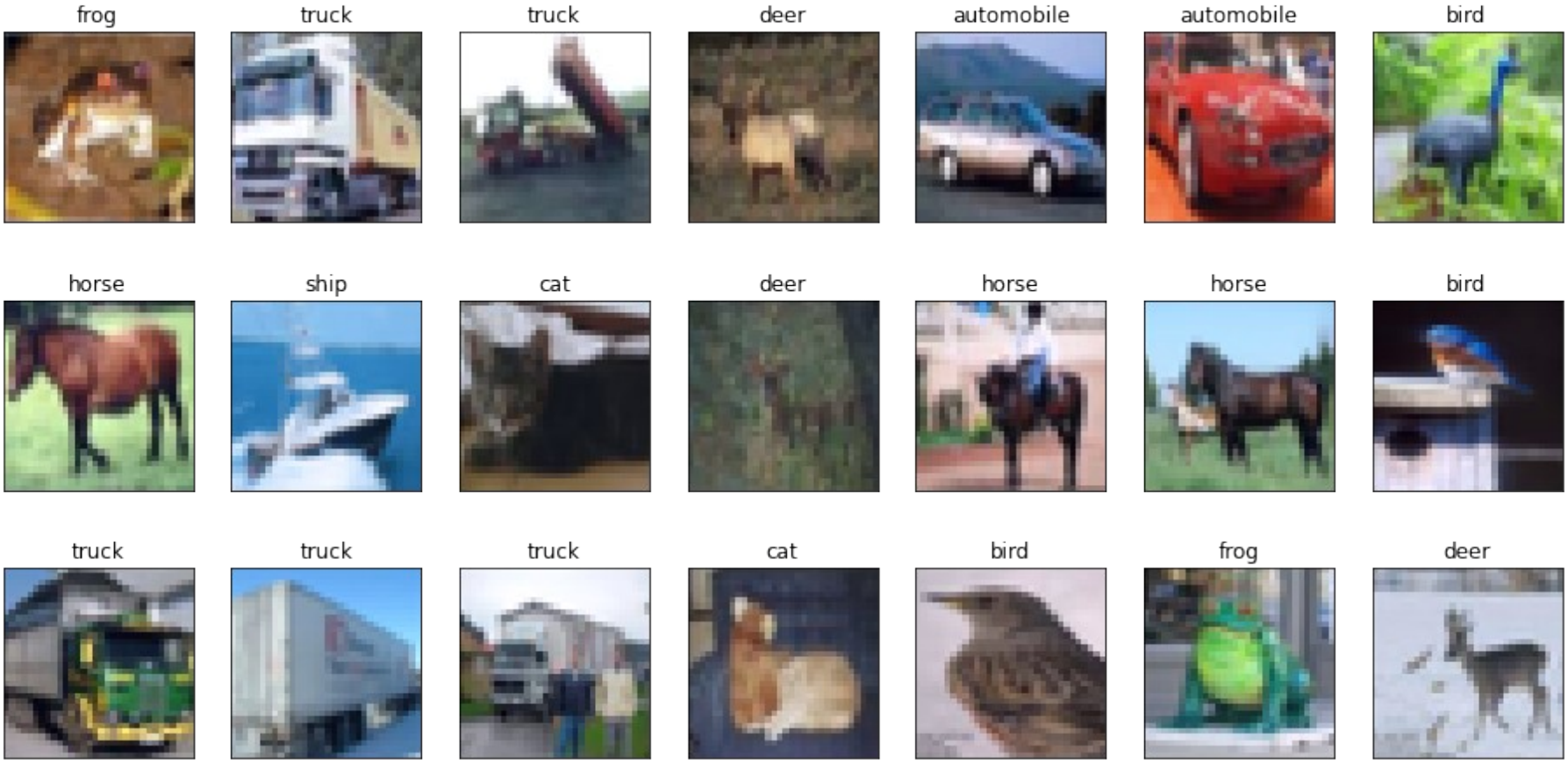
The models are very similar to those in original paper. First layer has a larger kernel (4*4) and stride 2, so the output of this layer is 16 by 16 images with 4 channels. All the other convolutional layers has kernel size 3*3 and padding 1 to preserve the size of convolved images. First layer has 4 channels and multiplied by two for some next layers.

The penultimate layer is average pooling and the last one is fully connected with soft-max activation function. Other layers has ReLu activation function.



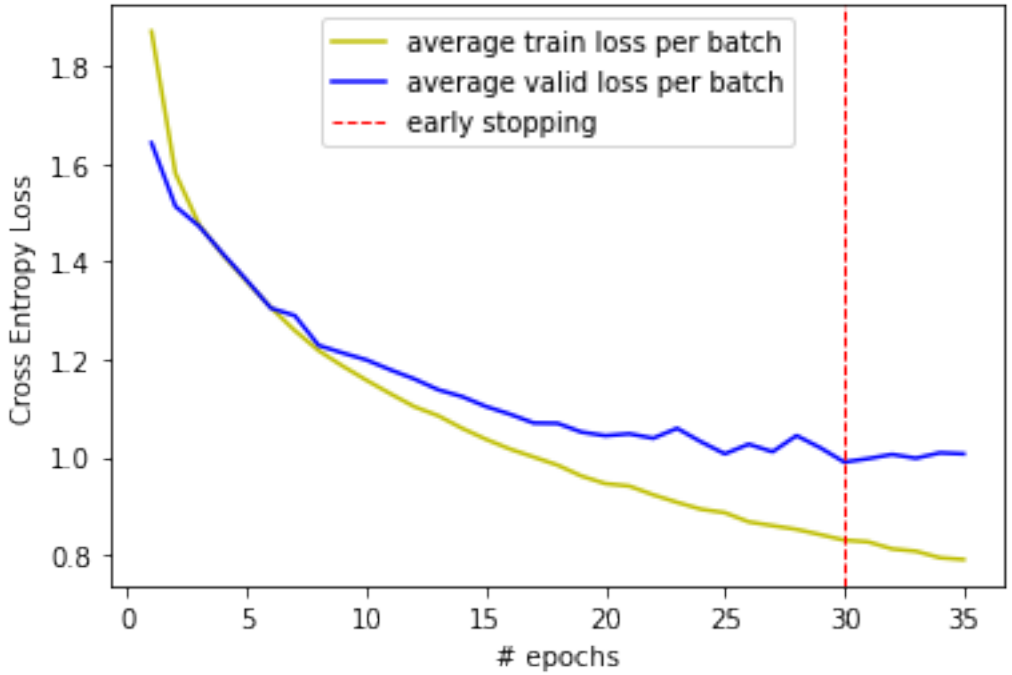
Data

I used a common data set CIFAR10, there is 10 thousand test samples and 50 thousand training samples. It has 10 different labels: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck . Each image is RGB 32 by 32 pixels.



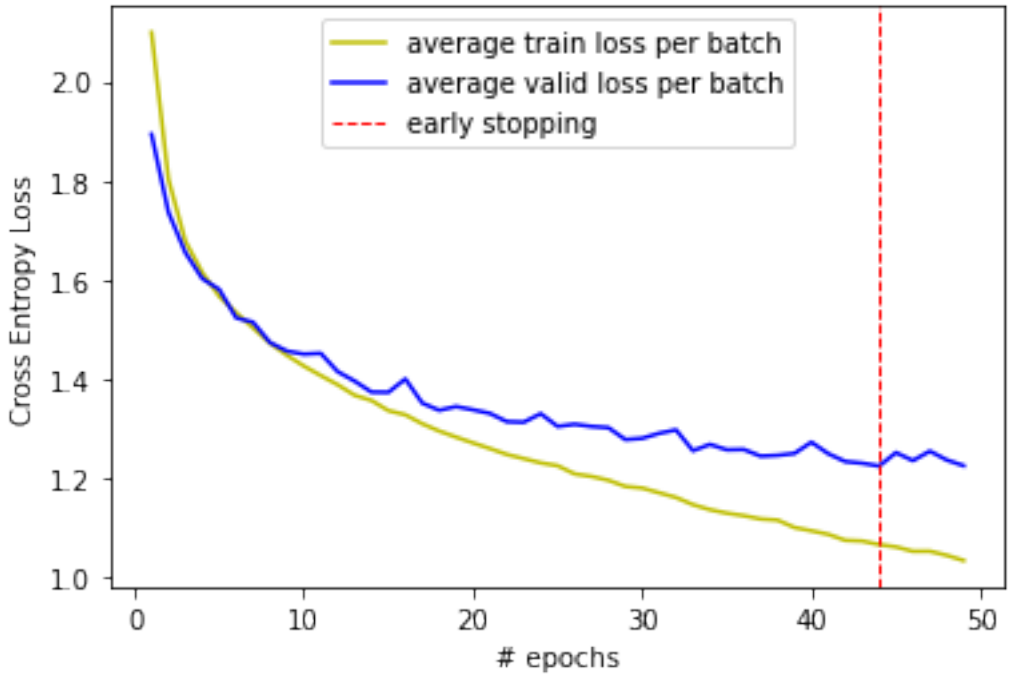
Training

ResNet12



I separated 20% of the training set to create a validation set. For training of all the networks I used an early stopping optimization technique with patience 5, the idea is to keep track of validation loss and save a model with minimal one, when the val loss doesn't reach its minimum in 5 next epochs the training stops and the final model is the one with minimal validation loss.

Plain12



For all the models I used Adam optimizer with learning rate 0.001.

As expected the ResNet12 converged faster. It was training for 30 epochs in contrary to 44 epochs for the plain model.

Also I trained a ResNet and a plain net with the same architecture but with tanh activation function instead of ReLU.

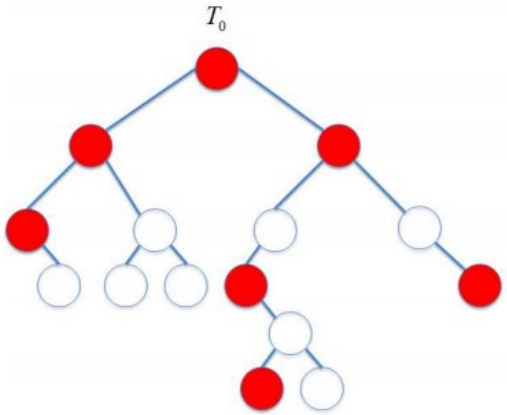
Performance

Confusion matrices, precision and recall are in the `nets_to_compurte_smoothness` notebook, also other experiments with architecture can be found there.

Accuracy on the test set:

ResNet12	65.63%
Plain12	57.40%
ResNet12 5 epochs	51.68%
Plain12 5 epochs	44.95%
ResNet12 with tanh activation	54.34%
Plain12 with tanh activation	51.93%

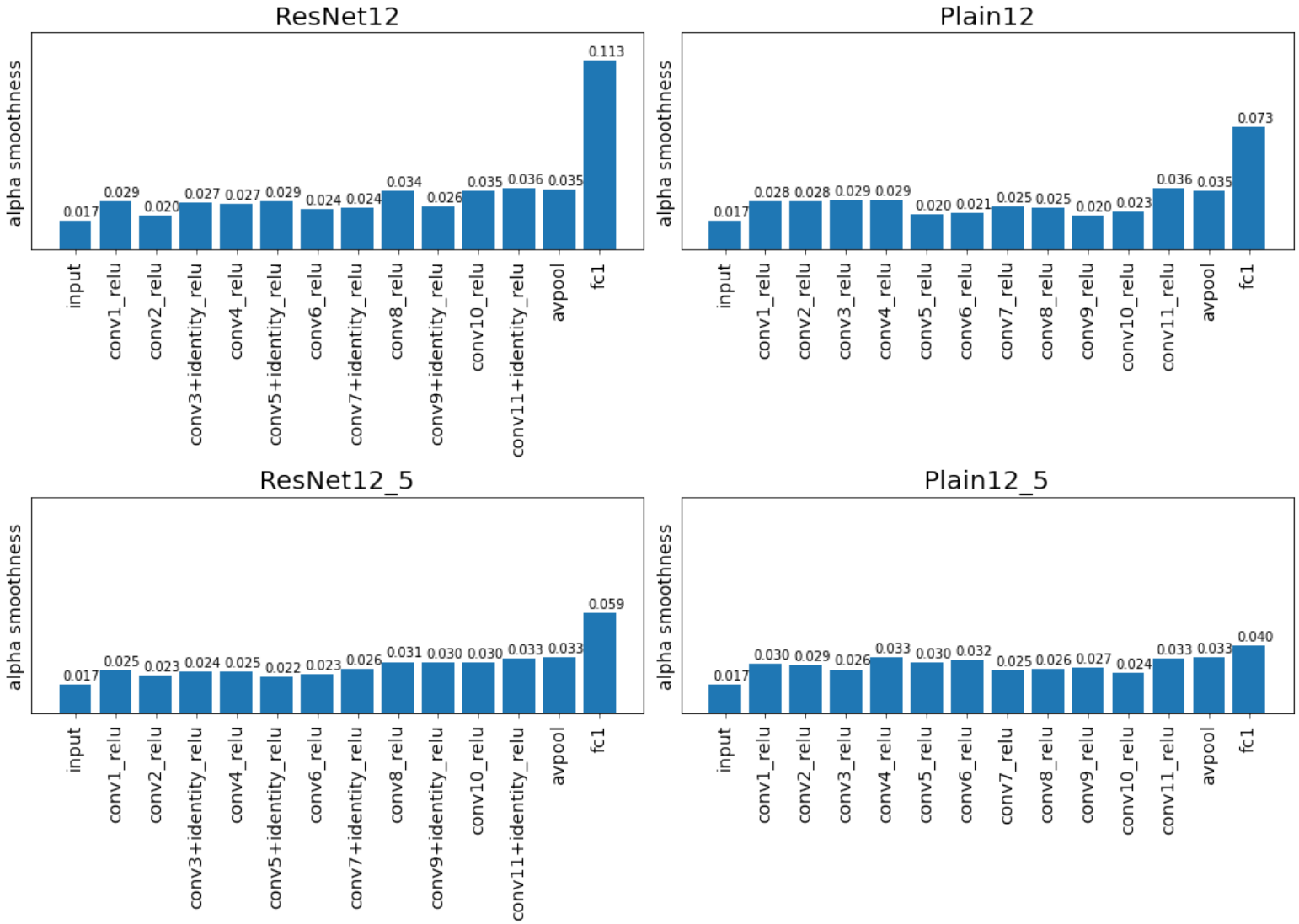
Besov smoothness



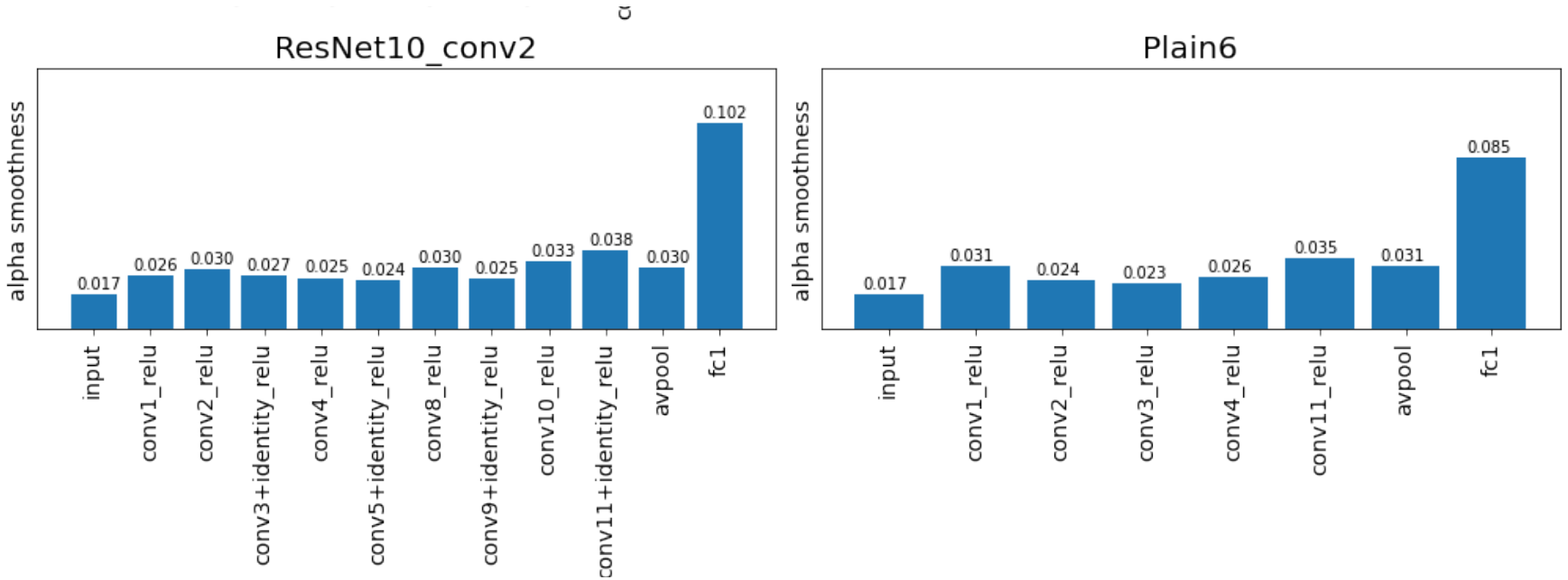
Neural networks have proved their efficiency, but still it is not clear what is going on inside the "black box". One approach is to use a function approximation theory to understand whether the geometric clustering improve with layers. For this we can calculate a weak type Besov smoothness for each layer. The idea is to use geometric wavelets. Since tao-sparsity of a tree is equivalent to Besov semi norm, we can use a Jackson estimate to evaluate the alpha.

I used 10 trees and square root of amount of features for each tree.

Besov smoothness for each layer of final ResNet12, Plain12 and their 5th epoch:



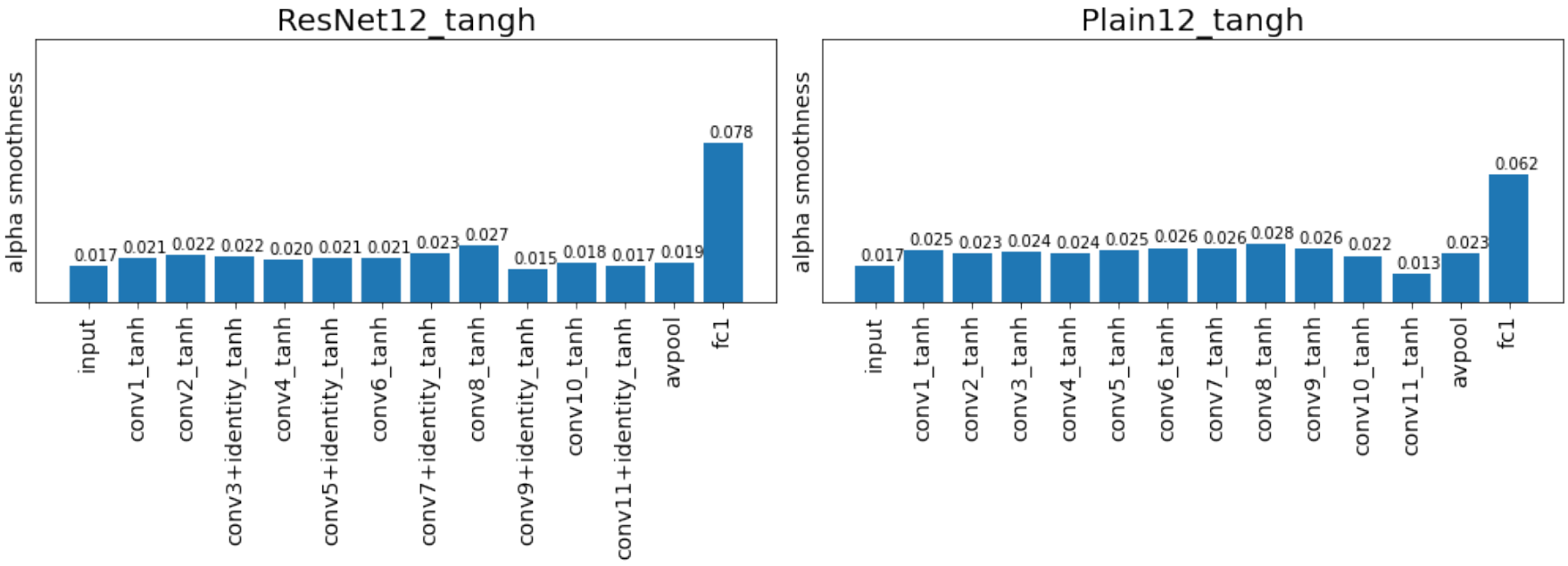
After performing the smoothness analysis, I changed the architecture of the networks. For ResNet I increased number of channels for second and third layer and deleted the block with layer 6 and 7. This network achieved 65.76 % of accuracy on the test data, in comparison to 65.63% of the previous model. For the Plain network I deleted 6 convolutional layers. The test accuracy is 61.78 %, in comparison to 57.40% for the previous model.



The smoothness of the average pooling layer
is smaller for these networks, so I tried to delete
this layer. But the test accuracy decreased

ResNet10	65.76 %	62.22 %	ResNet10 no AvPool
Plain6	61.78 %	57.76 %	Plain6 no AvPool

Smoothness analysis of ‘wrong’ networks with tang activation instead of ReLU:



According to this analysis I deleted layers

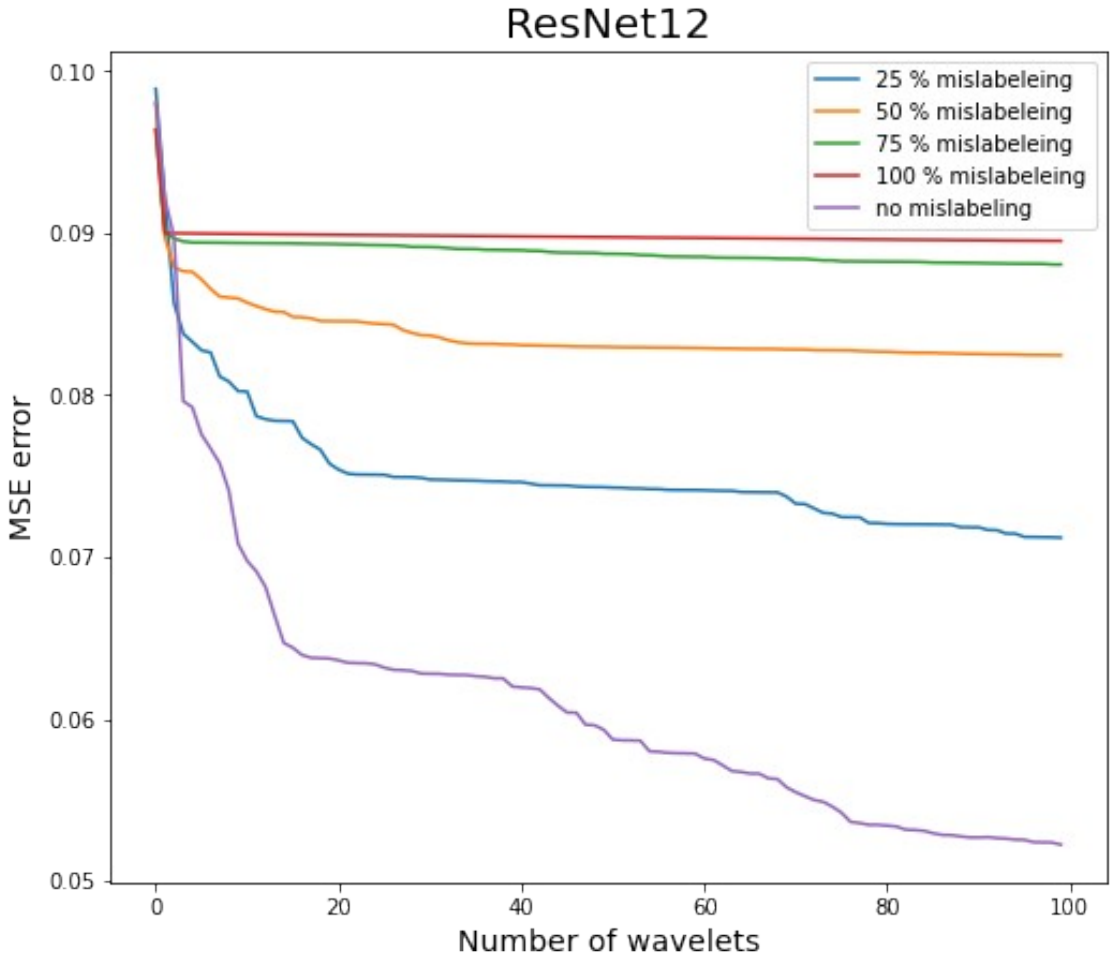
9, 10, 11 from both networks and got much smaller

networks that perform similarly on the test data.

Here we can see that relying on smoothness analysis
we can get a smaller network that performs almost
the same or even better than the initial one.

Initial network	**Test accuracy**	**Test accuracy**	**Modified network according to smoothness alnlysis**
ResNet12	65.63%	65.76 %	ResNet10
Plain12	57.40%	**61.78 %**	Plain6
ResNet12 with tanh activation	54.34%	53.69%	ResNet9 with tanh activation
Plain12 with tanh activation	51.93%	52.44%	Plain6 with tanh activation

Mislabeling



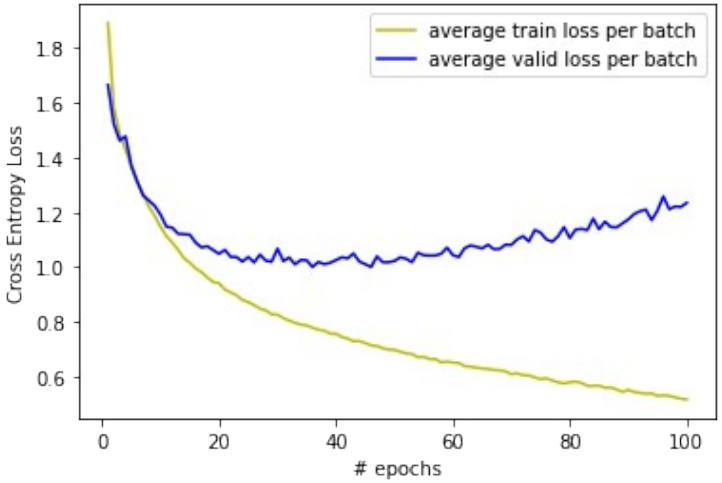
Besov smoothness on the last fully connected layer or ResNet12 with mislabeling:

No mislabeling	0.1134
25%	0.0468
50%	0.0206
75%	0.0079
Random lables	0.0046

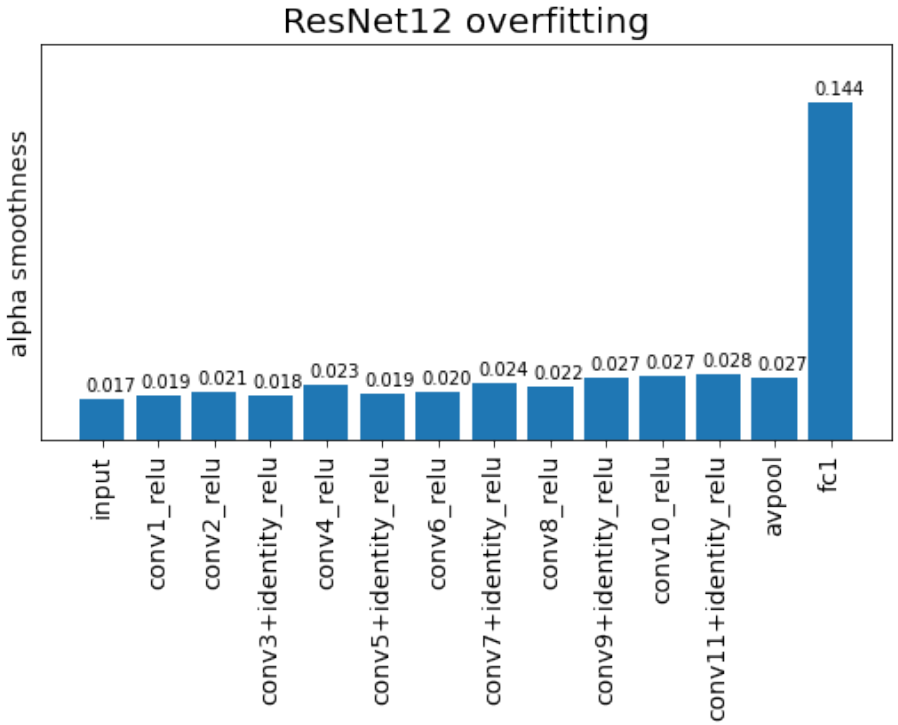
As expected, the amount of mislabeled data has a negative correlation with the alpha smoothness.

Smoothness of an overfitted model

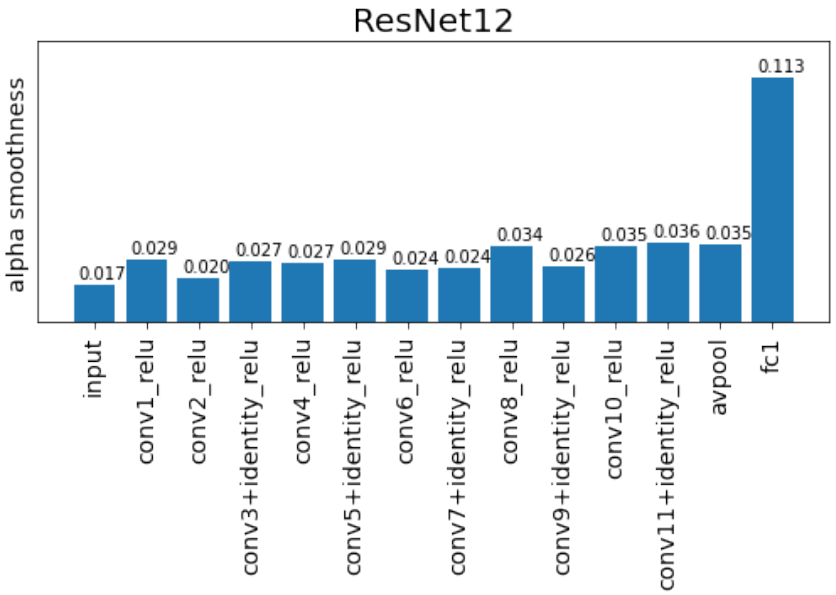
Training:



Alpha smoothness for overfitted ResNet12:



Smoothness for ResNet12:



Conclusions

In this project I trained a small ResNet and Plain one. I computed a weak type Besov smoothness of each layer of these networks using geometric wavelets and function space theory. We can see that the last layer smoothness strongly correlates with performance of a network. I expected to see alpha smoothness increasing gradually from layer to layer, but it is not improving much and only for the last layer it increases abruptly. I suppose it can relate to using fully convolutional NN.

Relying on smoothness analysis I changes some architectures – increased the number of channels or deleted some layers whose smoothness decreased. For some it had a positive effect on performance and reduced a complexity of a network, while for deleting average pooling layer it had a negative effect.

Also I presented mislabeled MS error of a forest and alpha smoothness of the last layer, and got similar results to the one in the paper.

In addition I performed a smoothness analysis for an overfitted ResNet12. We can see that the alpha smoothness for the overfitted model is mostly smaller for all the convolutional layers, but bigger for the last fully connected one.