Landmark Classification From Images Using Convolutional Deep Neural Networks

Buğra Aker Yılmaz
Aalto University
Espoo, Finland

Beyza Bütün
Aalto University
Espoo, Finland

***Abstract*—A set of experiments regarding Google's Landmark Recognition Challenge 2019[1] is presented here. They consist of experiments on data, on different network architectures and on hybrid models with some heuristics.**

***Index Terms*—Image Classification, Computer Vision, Deep Learning**

Code: https://github.com/akeryilmaz/DeepLearningProject

I. Introduction

Photographs are an inevitable parts of our vacations. However, when you go through these photographs after a while, it may be difficult to remember in which part of the city you were or what is the name of the building on the background. These questions are only a few of those which is tried to be answered by Landmark Recognition problem.

With the advent of Computer Vision, solving this problem does not seem to be impossible as before. However, there are still obstacles that are needed to be removed in the area. Google's Landmark Recognition Challenge[1] is organized for this reason. The goal of the challenge is to encourage researchers to study the obstacles and improve the performance of existing models or implement new solutions.

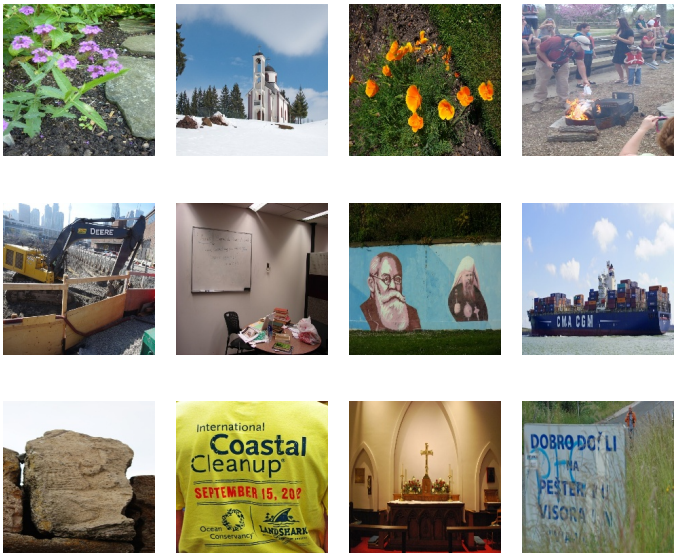


Fig. 1: Some image samples from dataset

II. Dataset

For the project, we have used the dataset provided by Google for Google Landmark Recognition Challenge 2019[1]. The dataset originally contains nearly 5 million images split into 3 sets, namely train, index and test, having landmarkID and URL for each image[2].

Competitors are allowed to download only the train set, until May 27, 2019. Therefore, we used only the train data. Train data contains approximately 4 million images from nearly 20000 landmarks, which are distributed highly unbalanced, 91,96% of the classes contain less than 50 images and 8,30% of the classes makes up 50% of all data.

The size of train set is 500GB in its row state. We have downloaded all data. After resizing all images to 224x224, we ended up having nearly 100GB of data. Since, this data is extremely big for having a reasonably fast model, we have used classes only which contain at least 1000 instances, which corresponds to top-47 classes, with 2 GB of images.

Moreover, the dataset is too noisy. First, images have very different width/height ratios. Second and more importantly, although all images are from the landmarks, some fail to represent the landmarks themselves. Some examples for this kind of instances are images of artworks that are taken inside of a museum or close-up images of plants and animals in a national park. Third, Even if the image shows outside of a building or a recognizable view, they are typically taken from very different angles.

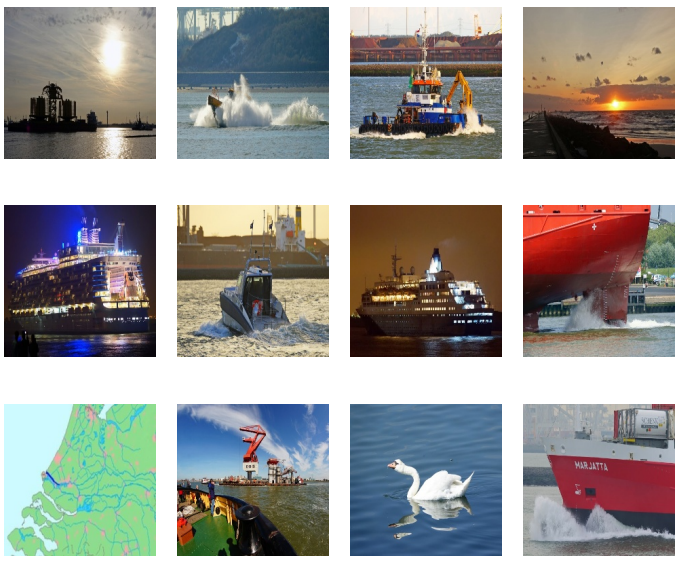


Fig. 2: Some image samples from same landmark (ID: 194914)

III. Evaluation

We have used Global Average Precision metric (GAP) known as micro Average Precision (microAP) and accuracy in order to evaluate our models.

If a submission has N predictions (label/confidence pairs) sorted in descending order by their confidence scores, then the Global Average Precision is computed as[3]:

$$\text{GAP} = \frac{1}{M} \sum_{i=1}^{N} P(i) rel(i)$$

where:

- N is the total number of predictions returned by the system, across all queries
- M is the total number of queries with at least one landmark from the training set visible in it (note that some queries may not depict landmarks)
- P(i) is the precision at rank i
- rel(i) denotes the relevance of prediction i: it's 1 if the ith prediction is correct, and 0 otherwise

The difference between GAP and accuracy is that GAP rewards true predictions with high confidence and penalizes false predictions with high confidence. That is to say, confidence matters in GAP.

IV. Methodology

We had 3 three different methods. The first method was classifying the images directly to landmarks. The second consisted of two phases, in which the close-up and indoor images are removed from the train sets of models and the models were trained on "clean" data. The last method was to give confidence penalties to indoor images.

We used different proposed architectures for classifying landmarks, namely VGG16, VGG19, Resnet18, Resnet50, Resnet101, Densenet121 and Densenet169.

A. VGG[8]

VGG models are one of the old types of Convolutional Networks. VGG16 and VGG19 are two deepest and best-performing models among all VGGs. They use smaller kernel size and deeper layers than AlexNet.

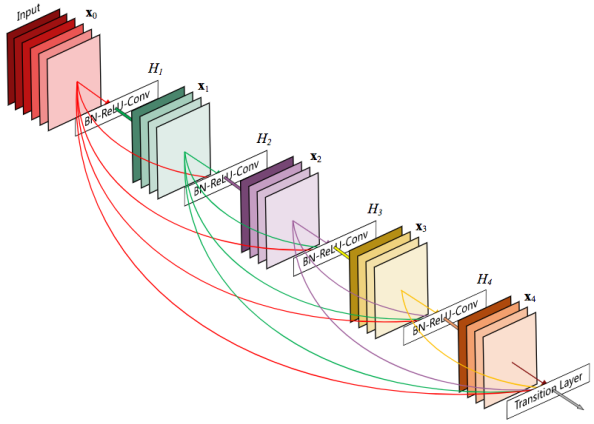
B. Resnet[7]

Resnets consist of blocks, and these blocks are made up of residual blocks. "In a network with residual blocks, each layers feed into next layer and also into the layers 2-3 hops away"[9]. Resnet models are named in terms of the number of layers they have.

C. Densenet[5]

Out of these 3 main architectures, Densenets are the ones that we did not mention in the lectures. Densenets are highly influenced by Resnets. In Resnets, the idea is to invent some by-pass connections, so that "the gradient can flow directly through the identity function from later layers to the earlier layers." [4]. In Densenets, this idea is generalized and by-pass connections are invented from an earlier block to all later blocks in a Dense Block.

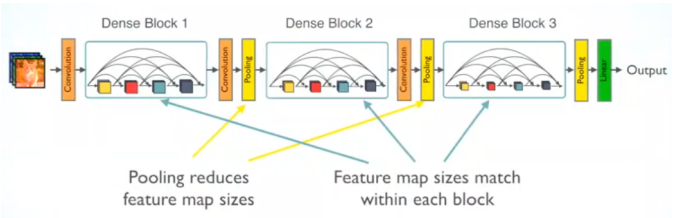
Fig. 3: A Dense Block Layout [5]



In between Dense Block are so called "transition blocks" that consist of a convolutional and a pooling layer that to enable down-sampling.

For instance, Densenet121 consist of 4 Dense Blocks and 3 transition layers in between blocks. Before moving to the first Dense Block, the model has a Convolutional layer, followed by Normalization, ReLu and Max-Pool layers. Dense Blocks have 6, 12, 24 and 16 Dense Layers, respectively. All Dense Layers in a block are connected to previous Dense Layers in the same block, in addition to being connected to the input of the block. Each Dense Layer consist of two layers of Normalization, ReLu and Convolutional layer. The last Dense Block is followed by a Normalization layer and a Linear Classifier.

Fig. 4: A Dense Net Layout [6]



D. Indoor-Outdoor Classification

For indoor-outdoor classifier, we used an already trained VGG16 on 1.8 million images from 365 scene categories, some of which are indoors and some which are outdoors [5]. We classified an image as indoor if it has at least a value of 85% as sum of confidence levels for indoor classes. For better understanding, the pseudo-code we used for this classification is given below.

```
Ifindoor (image):
confidences, classes ← model(image)
indoorProb ← 0
outdoorProb ← 0
ifindoor ← False
while outdoorProb < 0.15 do
    currentClass ← classes.pop()
    if currentClass is indoors then
        indoorProb ← indoorProb + confidences.pop()
    else
        outdoorProb ← outdoorProb + confidences.pop()
    end
    if indoorProb > 0.85 then
        ifindoor ← True
        break
    end
end
return ifindoor
```

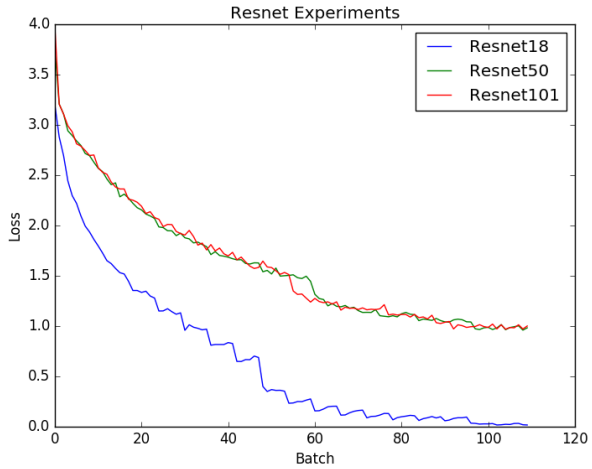
Our assumption for indoor images was that they do not contain useful information regarding landmark. We used this pretrained network as a binary classifier in order to increase the performances of our main networks mentioned above.

V. Experiments

A. First Experiments with Different Architectures

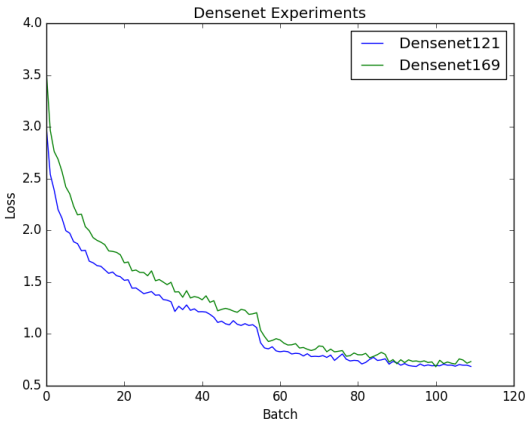
We started our experiments with trying out different architectures that are above mentioned. In all of our experiments for this section, we have used 7/8 of randomly shuffled data as train and 1/8 as test data. We used a batch size of 32. We normalized data before feeding to network. We used Cross Entropy Loss and Adam Optimizer. We modified all architectures such that the last linear layer's output was 47, which is equal to number of classes we used.

1) Resnets: We have experimented with Resnet18, Resnet50 and Resnet101. Learning rate was 0.01 and number of epochs was 10. The graph for these experiments are given below.



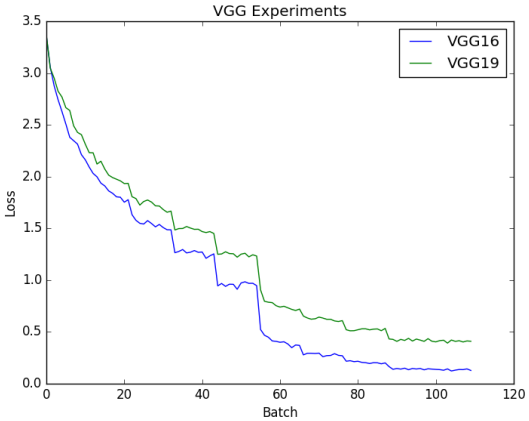
In the graph, we can see that Resnet50 and Resnet101 settle for a worse local minimum while Resnet18 could find a better local minimum. This is most likely due to model complexity. Deeper architectures such as Resnet50 and Resnet101 did not do well compared to Resnet18. Thus, we can conclude that Resnet18 has sufficient and "optimal" model capacity for the task.

2) Densenets: We have experimented with Densenet121 and Densenet169. Learning rate was 0.01 and number of epochs was 10. We multiplied learning rate with 0.1 at fifth and eighth epoch, since we observed that previous experiments had stall in sixth and ninth epochs.



In above graph, we see that the training performance of two networks are very similar compared to their model capacity. We can also see form the sharp decrease towards the middle of the graph that the interruption to learning rate helped especially at the fifth epoch.

3) VGGs: We have experimented with VGG16 and VGG19. Learning rate was 0.0001 and number of epochs was 10. We multiplied learning rate with 0.1 at fifth and eighth epoch.



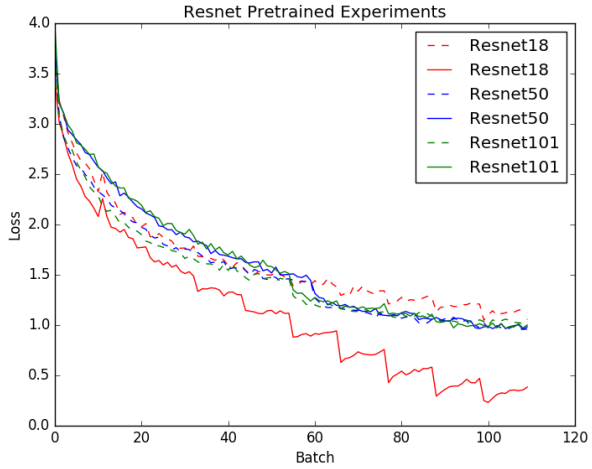
As in the Resnet graph, the deeper network has worse loss. Moreover, as in the Densenet graphs, we observe that the interruption to learning rate was successful.

B. Fine Tuning of Resnet18 and Densenet121

An overview of our experiments mentioned above is in the table below. We see that even though VGG models had the least loss levels, they generalized poorly. The most successful models were Resnet18 and Densenets. However, while comparing Densenet121 and Densenet169 we considered their model capacities and training times, and decided to continue with Densenet121.

MODEL	ACCURACY	LAST LOSS
Resnet18	0.741	0.276
Resnet50	0.684	0.945
Resnet101	0.677	1.001
Densenet121	0.749	0.686
Densenet169	0.741	0.733
VGG16	0.666	0.128
VGG19	0.644	0.410

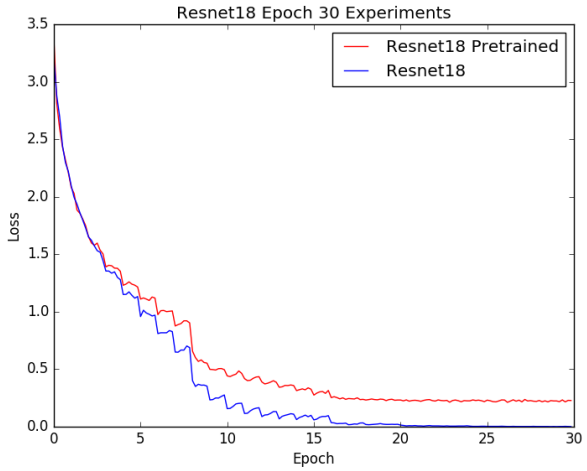
1) Pretrained Model vs. Randomly Initialized Weights: We experimented with Resnet18, Resnet50 and Resnet101 for this section. We did not try this on Densenet considering the unsuccessful results and long training time of Densenet.

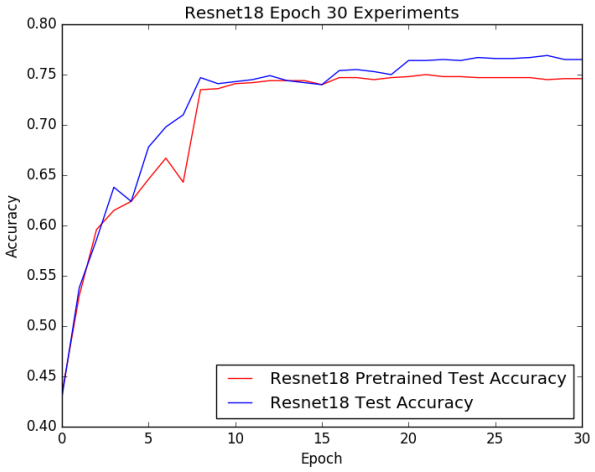


From the graph above, we see that there is not much difference between pretrained and randomly initialized models, except Resnet18. However, the accuracy and loss table below shows that in Resnet18, there is an overfitting of randomly initialized model Resnet18. Furthermore, there is no significant difference between the accuracies of the models, and even if there is, randomly initialized models outperformed pretrained ones.

MODEL	ACCURACY	LAST LOSS
Resnet18	0.669	0.384
Pretrained	0.621	1.165
Resnet50	0.684	0.945
Pretrained	0.687	0.983
Resnet101	0.677	1.001
Pretrained	0.679	1.060

2) Increasing Number Of Epochs: We tried to train Resnet18 for longer epochs to make sure we do not stop early. We did not do this experiment for Densenet, since we were not satisfied with the results.





As can be seen in the two graphs above, there is only a slight change of both loss and accuracy between epochs ten and twenty. Doubling the training time, it had only an increase of 2% in accuracy for only one of our models. We did not make any other experiment with 30 or 20 epochs, but it is good to keep in mind this slight improvement for a final model.

C. Cross Validation of Resnet18 and Densenet121

We have validated Resnet18 and Densenet121 models with a 5-fold cross-validation. For both architectures, the data is shuffled randomly and five equal chunks of data has been created. The architectures are trained on 4/5 and tested on 1/5 of the data for a different chunk as test set at each time. There are 5 runs of both architectures. For each run accuracy and GAPs are calculated.

ACCURACY	RESNET18	DENSENET121
Accuracy1	0.659	0.738
Accuracy2	0.728	0.744
Accuracy3	0.733	0.744
Accuracy4	0.730	0.753
Accuracy5	0.719	0.749
Average Accuracy	0.725	0.746
Average GAP	0.666	0.693

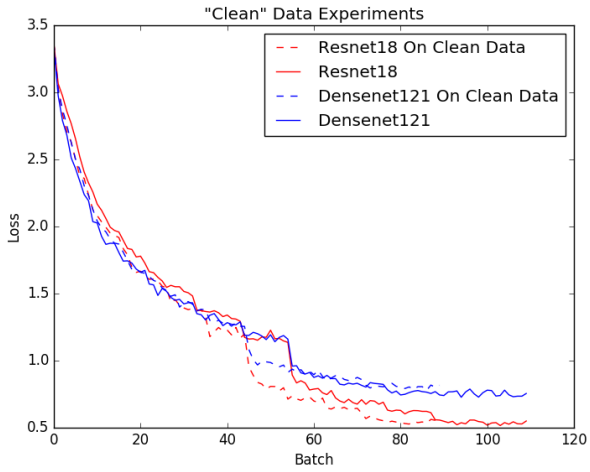
Outperforming Resnet18 in terms of test data prediction accuracy in average and on every run, Densenet121 is our most successful model.

D. Training on "Clean" Data

As previously mentioned, the dataset which we used is too noisy. Some images don't represent any landmark, even though they are actually from landmarks. Therefore, we have decided to split them into indoor and outdoor images in order to work with clean data. We split the data into training and test sets with a ratio of 1/8. We created two copies of these sets. Using the indoor-outdoor classifier we removed instances that are classified as indoor images from one of the copies. The other one stayed as it is. We trained two models for each of these copies, one with a train set of indoor images deleted. The models were Resnet18 and Densenet121, since they were the best models.

Fig. 5: Some image samples that are classified as indoor

For this part of our experiments, we used a batch size of 32. We normalized data before feeding to network. We used Cross Entropy Loss and Adam Optimizer. The models are tested on the same "noisy" data. The indoor-outdoor classifier is mentioned in "Methods" section of this paper. We used accuracy and GAP metrics on test data.



From the graph above, it can be said that there is not much of a difference between training process of two experiments. However, from the table below, it is evident that models containing indoor images did well on test data which contains indoor images. In conclusion, our assumption that indoor images do not contain useful information regarding landmarks has failed.

Nonetheless, it is important to note that in our case, the number of classes is too low compared to actual competition. It is possible that indoor images are useful for a small number of classes but not useful -maybe harmful- for bigger and sparser data. Moreover, it might be possible to train another classifier that distinguishes not indoor images but close-up images of objects.

MODEL	ACCURACY	LAST LOSS	GAP
Resnet18	0.727	0.550	0.687
Clean Data	0.706	0.540	0.662
Densenet121	0.737	0.755	0.694
Clean Data	0.714	0.816	0.666

E. Penalty on Indoor Images

So far we did no attempt to improve confidence of true classifications and penalize confidence of false classifications. As above mentioned in "Evaluation" section, confidence scores matter in GAP calculations. Thus, one way to improve GAP is to penalize the confidence of potentially false classifiable instances.

Our assumption in this section was that the indoor images were potentially false classifiable. Therefore, we applied a confidence penalty on them, as given below.

if *image is indoor* **then**
  | *confidence* $\leftarrow$ *confidence* $* 0.1$
end

In this part of our experiments, we used our trained Resnet18 and Densenet121 and the indoor-outdoor classifier that are mentioned in the subsection "Training on 'Clean' Data".

MODEL	GAP	GAP with Penalty
Resnet18	0.687	0.668
Clean Data	0.662	0.642
Densenet121	0.694	0.676
Clean Data	0.666	0.648

As the results above indicates the experiment have failed. This can be better explained by the table given below. The table is created on the test run of our Densenet121 model on the "noisy" data. The vast majority of falsely classified instances are outdoor images and nearly 2/3 of indoor images are classified correctly.

	Outdoor	Indoor
Classified True	7279	269
Classified False	2561	115

VI. Conclusion

In conclusion, we experimented on different architectures of Convolutional Deep Neural Networks, for the task defined by Google Landmark Recognition Challenge 2019[1]. Most successful 2 models are found out by training on 7/8 of total data and testing on 1/8. They are then fine-tuned and validated with 5-fold cross-validation. We experimented on training data to improve the results but were not successful. We experimented to combine two models using a heuristic to improve performance. Highest average GAP score we achieved with cross-validation is 0.693. The highest GAP on test set provided by Goggle is 0.21657 by the time this paper is written. However, it is good to keep in mind that the models for the challenge are trained with nearly 20 000 landmarks most of which contain very few instances. Also, the figures are obtained on a separate test data, whose relation to train data is unknown for us. Therefore, our problem that classifying images from 47 landmarks each of which has at least 1000 images is not equivalent to Google's competition, and comparing the figures is not so fair.

References

[1] Google Landmark Recognition Challenge 2019. Available at https://www.kaggle.com/c/landmark-recognition-2019/overview

[2] H. Noh, A. Araujo, J. Sim, T. Weyand, B. Han. 2017. *Large-Scale Image Retrieval with Attentive Deep Local Features*, Proc. ICCV'17. Dataset available at: https://github.com/cvdfoundation/google-landmark

[3] F. Perronnin, Y. Liu, and J.-M. Renders. 2019 *A Family of Contextual Measures of Similarity between Distributions with Application to Image Retrieval* Proc. CVPR

[4] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, A. Torralba. 2017. *Places: A 10 Million Image Database for Scene Recognition*. Available at: http://places2.csail.mit.edu/PAMI_places.pdf

[5] G. Huang, Z. Liu, L., Maaten. 2018. *Densely Connected Convolutional Networks* Available at: https://arxiv.org/pdf/1608.06993.pdf

[6] Image taken from: https://towardsdatascience.com/review-densenet-image-classification-b6631a8ef803

[7] K. He, X. Zhang, S. Ren, J. Sun. 2015. *Deep Residual Learning for Image Recognition*.

[8] K. Simonyan, A. Zisserman. 2014. *Very Deep Convolutional Networks for Large-Scale Image Recognition*.

[9] Quote from:https://towardsdatascience.com/residual-blocks-building-blocks-of-resnet-fd90ca15d6ec