Final Project – Deep Learning (097200), Technion

Subject: Image retrieval using binary hash codes with different transfer learning nets

Based on the paper:

Deep Learning of Binary Hash Codes for Fast Image Retrieval

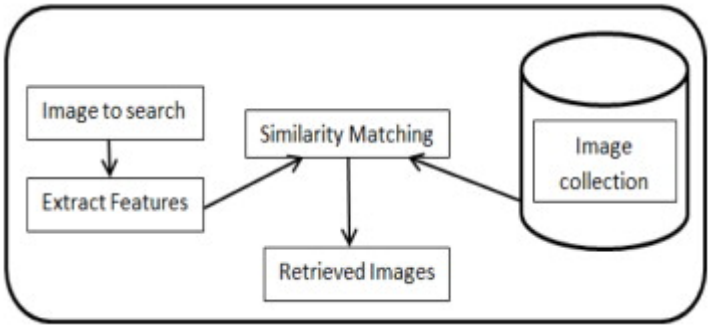
Kevin Lin, Huei-Fang Yang, Jen-Hao Hsiao and Chu-Song Chen

Problem Definition:

Given a query image, we would like to retrieve the most alike images. Our final retrieval list is ranked, meaning that the first image is the most similar to the query image and so on.

Figure 1 shows a simple scheme of the image retrieval process.

Figure 1. General image retrieval process



Solution:

My suggested solution for how to retrieve the images is based on the paper: Deep Learning of Binary Hash Codes for Fast Image Retrieval. The paper was presented in cvprw15.

In the paper, they suggest adding a latent layer to the end of the net with h nodes (called bits) and learn the weights of the nets as usual. After that we transform the nodes scores (weights) of the latent layer to a binary score using a threshold (if the score is grater then the threshold, assign 1, otherwise assign 0).

Finally, we represent each image using the h bits (ones and zeros) and then we rank the images by the number of similar bits (the more similar, the higher the image is ranked). An example of this process is shown in figure 9.

In the paper, they used AlexNet as a transfer learning net. AlexNet is a net that was trained on ImageNet dataset, this enormous dataset contains approximately 1.2M images. In my project I'm analyzing the differences of using different transfer learning nets, and not only AlexNet. In Figure 8, you can see a general scheme of the training process with a general transfer learning net.

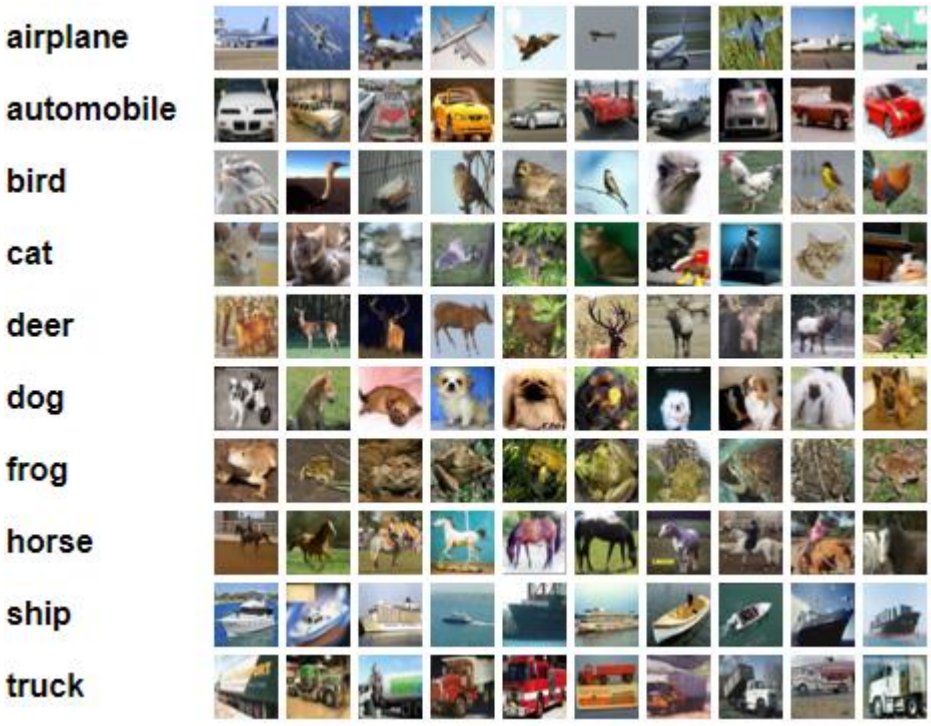
Datasets:

As mentioned, the transfer learning net was trained on **ImageNet**, this dataset contains about 1.2M, 256X256 px images (after preprocessing).

To train the net and test it, I used **CIFAR10** dataset, this dataset contains 60K, 32X32 px images in 10 different classes, exactly 6,000 images per class. The dataset is split in to 50K images for training and 10K for testing. In Figure 2 you can see an example of the CIFAR10 dataset.

CIFAR10 was chosen mainly because it was used also in the original paper and since we are using complicated nets in the transfer learning stage, the dataset needs to contain small images in order to achieve good results.

Figure 2. CIFAR10 Example



Transfer learning nets:

In addition to AlexNet, I run the model using GoogLeNet (also called inception v1), ResNet18, VGG16 and VGG19 as transfer learning net.

In Figures 3-7, you can see the architecture differences between these nets.

Figure 3. VGG19 architecture

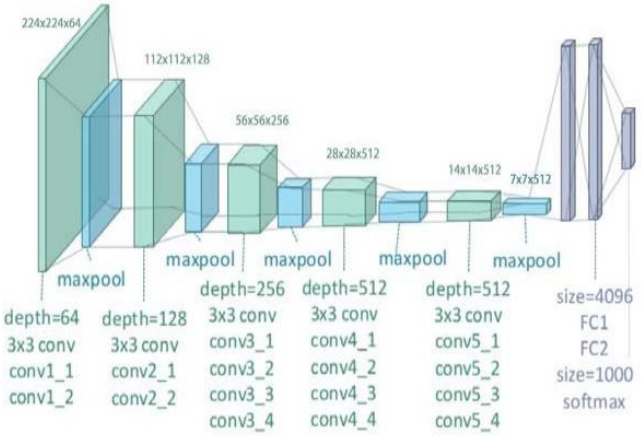


Figure 4. VGG16 architecture

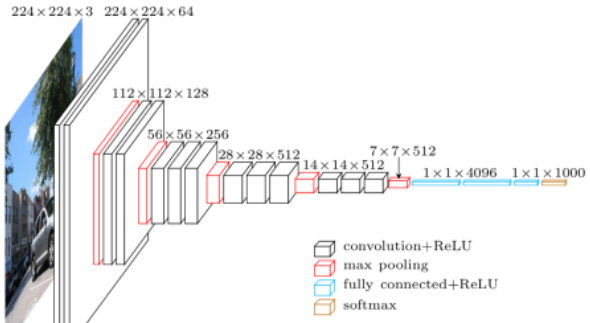


Figure 5. GoogLeNet architecture

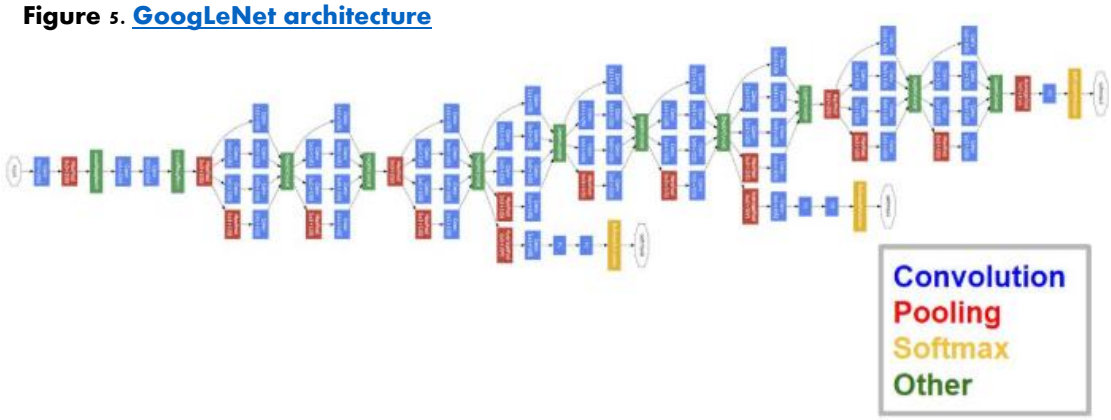


Figure 6. ResNet18 architecture

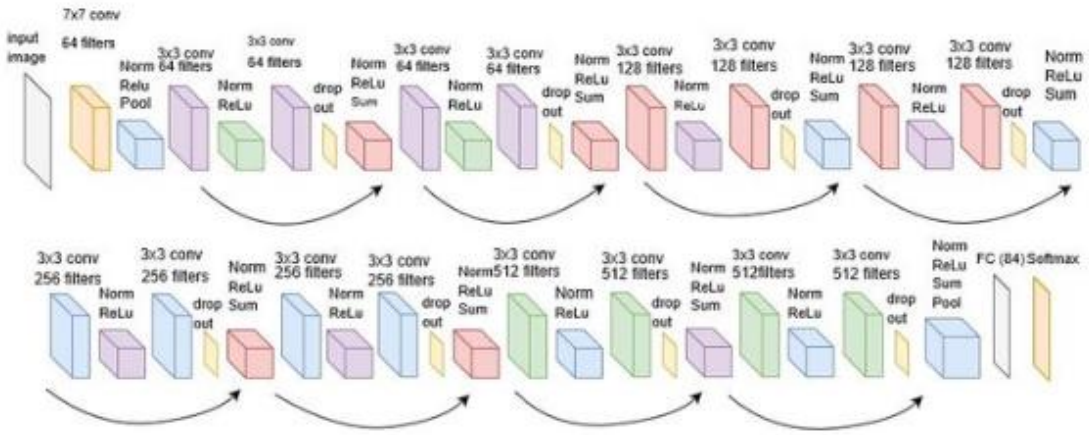
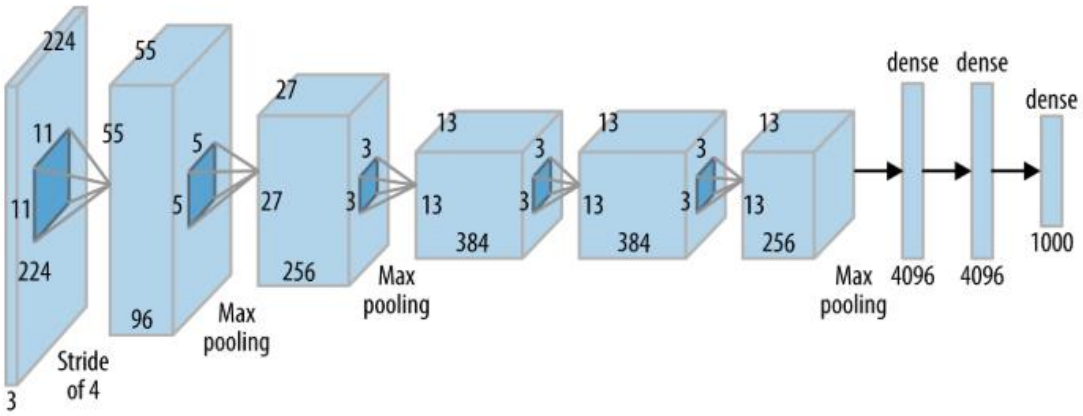


Figure 7. [AlexNet architecture]



Implementation and experiments details

Data Augmentation:

To avoid over-fitting and to generalize the training and achieve better results on the test set, I used the following data augmentation:

On the train set I resize the image features into 256 and then I used random crop to 227 and random horizonal flip.

On the test set resize the image features to 227.

I normalized both sets with respect to the mean and std of the train set.

Training Procedure's Hyper Parameters:

All models where trained using 50 epochs and a learning rate of 0.01

Optimizer:

SGD is the selected optimizer with momentum of 0.9 and weight decay of 0.0005

Batch size:

The batch size has a huge influence on the accuracy but since we are dealing with huge nets, the size is bounded because of the GPU's limited capacity. For that reason, we run the experiments on two batch size setups, the first is the maximum batch size possible using the transfer learning net (from [200,150,100,75,50,40,30,20,10]), and the second is the maximum batch size (from the same list) that is possible to all transfer learning nets.

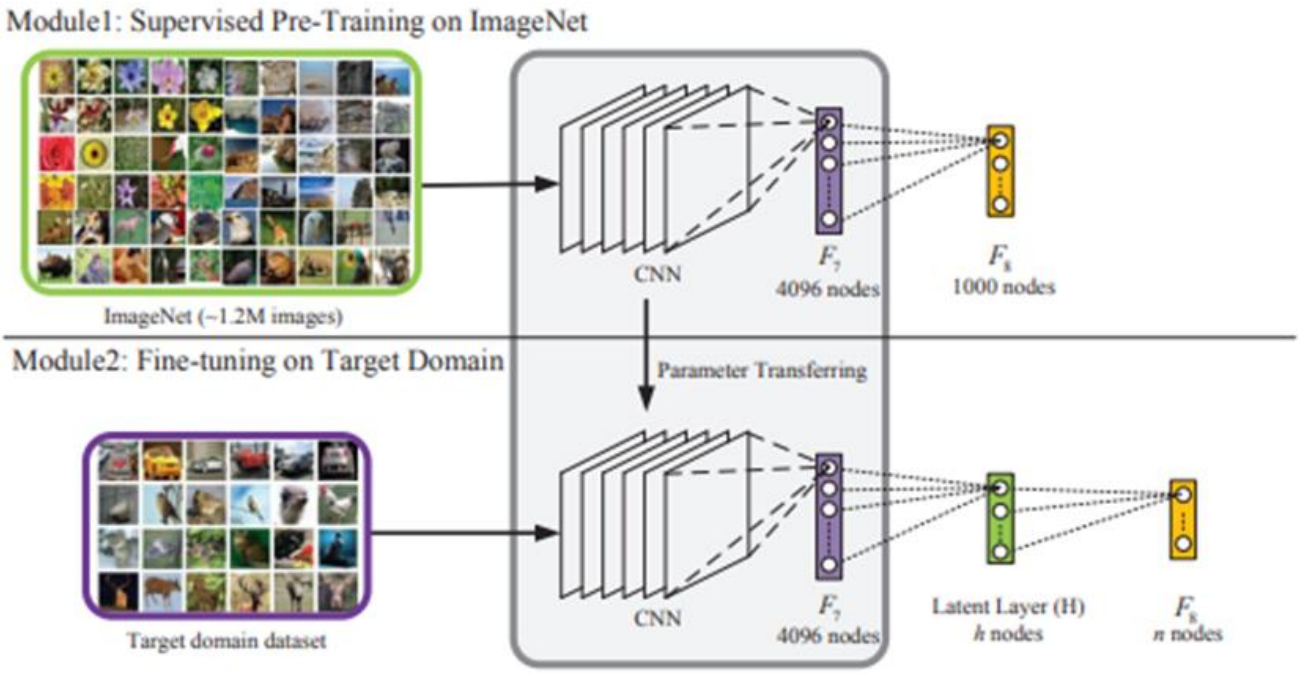
So, the batch sizes for each transfer learning net are:

```
{'AlexNet':[30,200],
 'VGG19':[30],
 'VGG16': [30,40],
 'ResNet18':[30,150],
 'GoogLeNet': [30,75]}
```

Model Architecture:

In general, my model architecture is based on the paper's architecture. A scheme of the architecture is shown in Figure 8.

Figure 8. **Train the model with transfer learning scheme**



Between the latent layer to the last layer (F8), we apply sigmoid as a transfer function. The last layer is a linear (fully connected) layer from h bits to 10 (n = number of classes).

In the training procedure, the weights of the transfer learning nets are tuned as well.

Image Retrieval Process:

For a given query image from the test set, the retrieval model needs to rank the 50K images from the train set according to the similarity to the query image. In the paper, they show the results on randomly 1000 query images but in my project, I'll show the results according to all query images (10K).

Figure 9. [Image Retrieval via Hierarchical Deep Search]

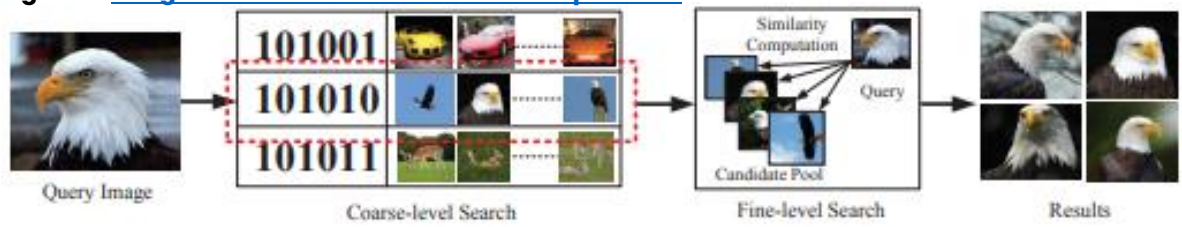


Figure 9 shows an example of the image retrieval using hash codes.

Image Retrieval Hyper Parameters:

The number of bits was set to 48

The threshold used for the binary bits is 0.5. That means that if a node had a score smaller then 0.5, the bit was set to zero, otherwise it was set to one.

Image Retrieval Measurements (based on Wikipedia):

Precision at k is the fraction of the top k objects that are relevant to the query.

$$Precision@k = \frac{\sum_{i=1}^{k} Rel(i)}{k}$$

Where Rel(i) is a binary score, 1 if the i-th image have the same label as the query image and 0 otherwise.

Mean average precision for a set of queries is the mean of the average precision scores for each query.

$$\mathrm{MAP} = \frac{\sum_{q=1}^{Q} \mathrm{AveP(q)}}{Q}$$

Average precision is:

$$\mathrm{AveP} = \frac{\sum_{k=1}^{n} (P(k) \times \mathrm{rel}(k))}{\text{Number of relevant images}}$$

P(k) is the Precision@k score and rel(k) is the same as above

Analysis and Results

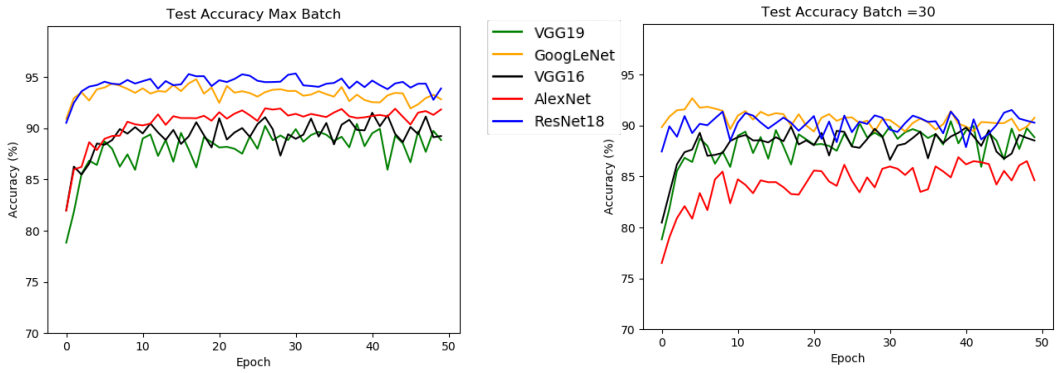
As mentioned, I ran the experiments with 2 setups, the first is with the maximal batch possible to each transfer learning net (from [200,150,100,75,50,40,30,20,10]), called 'Max Batch' and the other is

with the batch that is joint to all transfer learning nets, in our case this batch size is equal to 30.

Training Analysis:

In Figure 10 you can see the analysis made on the trained net using different transfer learning net. The main reason of this analysis is to see if there is a connection between the test set accuracy to the image retrieval scores.

Figure 10. Accuracy (over epochs) on the test set for every transfer learning net



In Figure 10 You can see that on both batch sizes, GoogLeNet and ResNet18 gets the highest accuracy. In addition, AlexNet gets the best improvement between the two batch sizes. About the VGG nets, we can see that there is not much difference between VGG16 and VGG19 when we examine the test accuracy.

Image Retrieval Live Examples:

Figure 11 you can shows an example of top-5 retrieval results when using different transfer learning nets and in figure 12 you can see the labels histogram when retrieving the top 1000 images.

Figure 11. Example of top-5 retrieved images when we used the max batch size

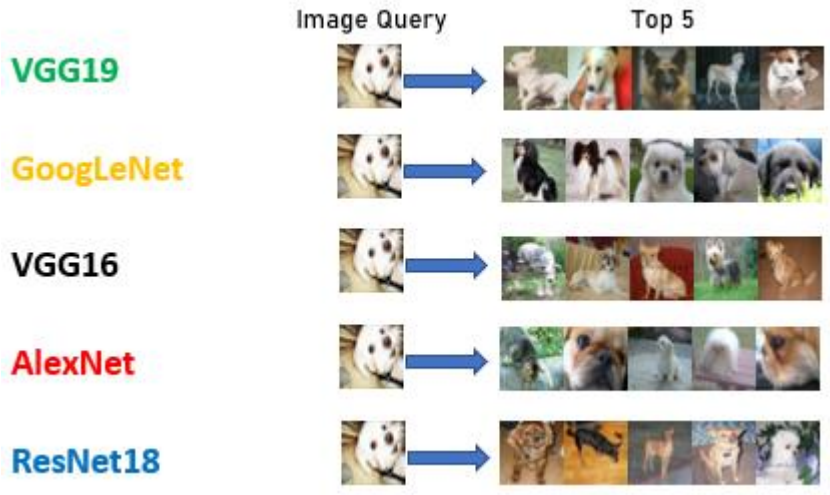


Figure 12. Labels histogram of the top 1000 retrieved images when batch = 30.

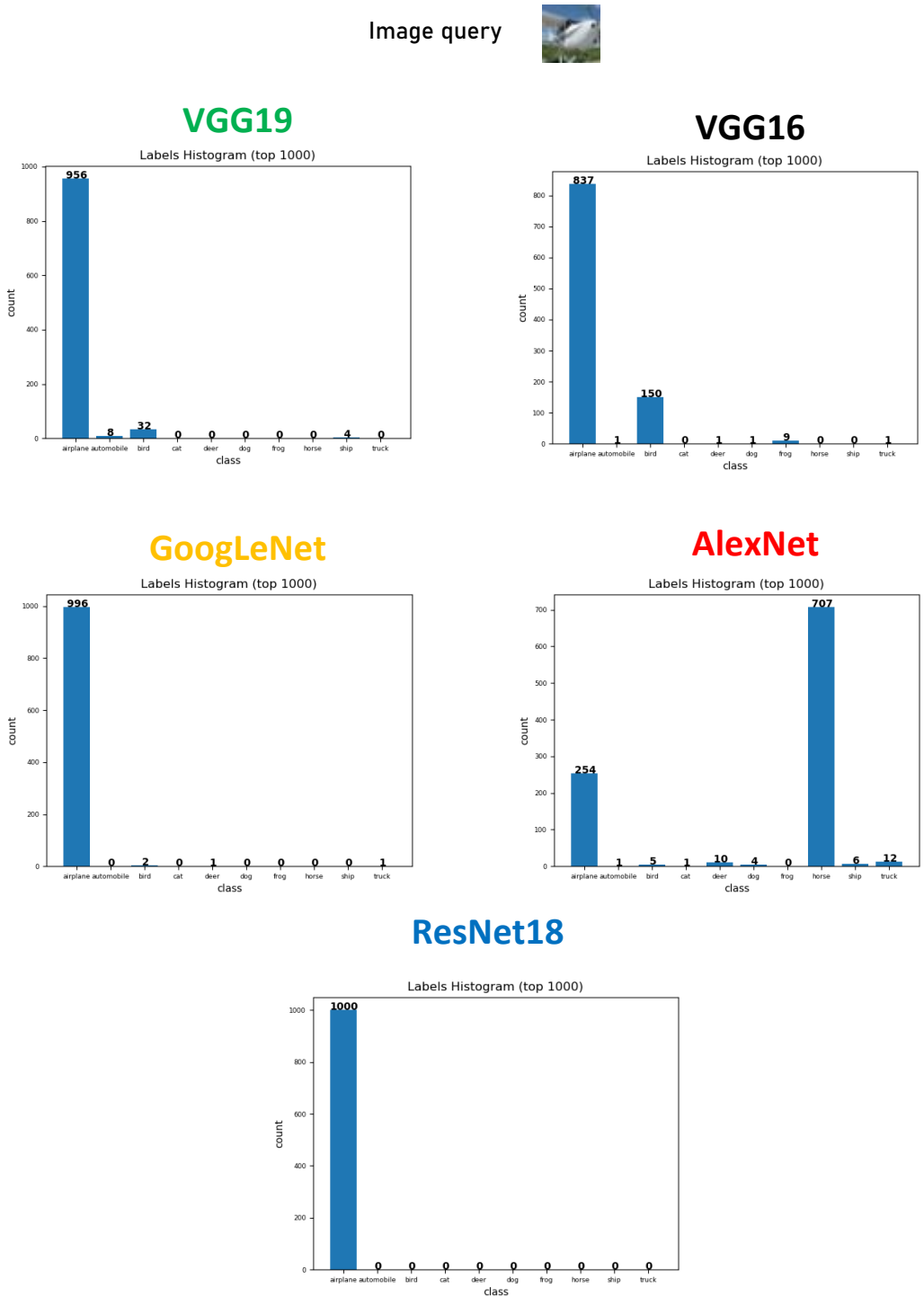


Image Retrieval Results and Analysis:

To be able to compare the scores, I took the top 1000 images retrieved for the MAP and Precision@k computation since the test accuracy was calculated over 1000 images as well. In figure 13 you can see that the Precision@k score is very solid for every value of k and for all nets (exclusive of AlexNet on batch size equal to 30). We can conclude that

the ratio between the relevant and non-relevant images is almost the same in every cutoff. As we saw in figure 10, the batch size has a huge impact on the Image retrieval results as well, in figures 13 and 14 we have another evidence for that.

Figure 13. Precision@k scores for every k

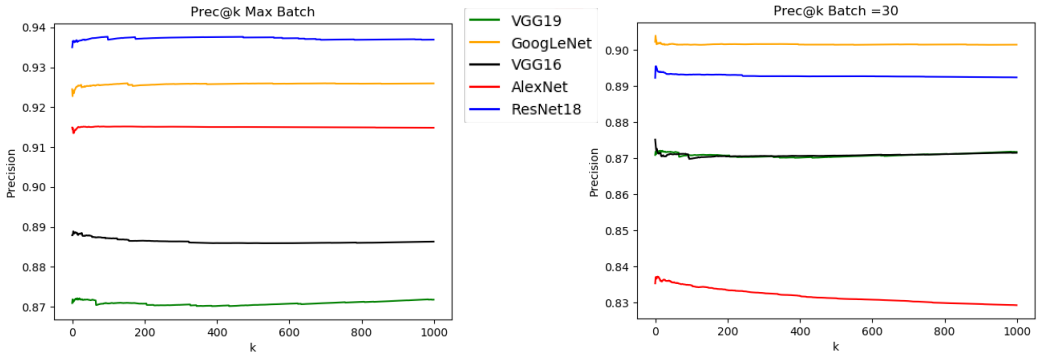


Figure 14. MAP scores and the improvement between the batch sizes

	AlexNet	GoogLeNet	ResNet18	VGG16	VGG19
Batch = 30	0.8103	0.9011	0.8912	0.869	0.8712
Max Batch	0.9299	0.9332	0.9477	0.8927	0.8712
Improvment	14.76%	3.56%	6.34%	2.73%	0.00%

After I saw that the batch size affects the image retrieval scores, I decided to examine if there is a connection between the test set accuracy to the image retrieval results. To do so, I compared in Figure 15 the test accuracy score to the MAP score. We can see that improving the test accuracy improves the MAP score as well but not in the same proportion. If we compare figure 13 and figure 14 to figure 10 you can see that the ranks of the transfer nets are almost the same (on batch equals to 30 and on max batch), this means that we can assume that if we improve the test accuracy we will probably get better retrieval scores.

Figure 15. MAP scores in relation to the accuracy scores on both batch sizes. MAP and test accuracy are both calculate over 1000 images.

	AlexNet	GoogLeNet	ResNet18	VGG16	VGG19
Test Accuracy	0.8463	0.9076	0.9028	0.8853	0.8885
Batch = 30	0.8103	0.9011	0.8912	0.869	0.8712
Improvment	-4.25%	-0.72%	-1.28%	-1.84%	-1.95%

	AlexNet	GoogLeNet	ResNet18	VGG16	VGG19
Test Accuracy	0.9184	0.9283	0.9389	0.8924	0.8885
Max Batch	0.9299	0.9332	0.9477	0.8927	0.8712
Improvment	1.25%	0.53%	0.94%	0.03%	-1.95%

Summary:

In this project I've implemented the model presented in the Deep Learning of Binary Hash Codes for Fast Image Retrieval paper.

In addition to the model presented in the paper, I've Implemented the model using several transfer learning nets and then I examined the differences between the scores and the linkage between the test set accuracy to the image retrieval scores.

Running Instructions:

1. Net training:
   To train the net, you need to run the following command:
   python train.py --name [model name] --lr [learning rate] --momentum [momentum] --epoch [number of epochs] --batch [batch size] --bits [number of bits] --path [folder path] --mid [transfer learning net]
2. Run Image Retrieval:
   To run the retrieval, you need to run the following command:
   python run_retrieval.py --batch [batch size] --bits [number of bits] --path [folder path] --mid [transfer learning net] --rand [number of random images per class]

Default values:

[model name] = 'my_model.pkl'

[learning rate] = 0.01

[momentum] = 0.9

[number of epochs] = 50

[batch size] = 200

[number of bits] = 48

[folder path] = ''

[transfer learning net] = 'AlexNet'

[number of random images per class] = 0, means all images

Remarks:

[transfer learning net] needs to be one of the following:

{'AlexNet', 'VGG19','VGG16','ResNet18','GoogLeNet'}