COGS 260 Final Project: Image Inpainting and Classifying

Lianming Shi
Electrical and Computer Engineering
PID: A99097650
l5shi@eng.ucsd.edu

Shengqi Feng
Electrical and Computer Engineering
PID: A53252866
shf060@eng.ucsd.edu

Abstract

This project focuses on solving the classifying problem with blocked part. Mean process of the project includes image inpainting, image deblurring and image classifying. The mean tool we implemented is convolution neural network such as VGG16. The goal of this project was to try different convolution network to improve our image quality to improve classifying ability of our origin classifying model. As a result, our model can do great inpainting on a series of feline image data and improved classifying accuracy about 10% compared to image with block, almost reach the accuracy of the original images.

1 Introduction

CNN (convolutional neural network) shows great power in classifying different images. With different constructions, the network shows different model size, accuracy, and efficiency in different topics. Typically, we have networks include AlexNet, VGGNet, which vary in number of hidden layers, combination in convolution layers and pooling layers.

There are many reasons may cause the damage to the origin image (get blocked part for origin image), including signal losing, physical barrel, watermark, etc. The goal of inpainting is to fill in a portion of an image that is either corrupted or unwanted. The applications of inpainting include: removing noise from an image, undoing deterioration, or to simply add or remove elements from an image. It can also be incorporated indirectly in compression wherein some percentage of original image can be transmitted and the whole can be reconstructed on the other end using a pretrained neural network.

There are many methods in inpainting problem, since aiming at classifying, we don't have to recover a perfect image, improving classifying accuracy is more important for us. In our project, we implemented an CNN that proved efficient in image deblurring. We just want to test whether similar layer structure convolutional neural network can have good effect in inpainting and image recovering. Once our first model can recover a blurred image, we can use similar construction to deblur these images.

In this project, we aim at inpainting the blocked feline images and making classification on feline with feline images after inpainting. We implemented two convolutional neural networks to finish inpainting and one convolutional neural network in classifying. We tested on different kind of blocking and model trained on different dataset to compare how the models work in different situation.

2 Method

2.1 Datasets

In our project, we choose feline images from ImageNet as our data. Feline is a lineage of carnivorans colloquially referred to as cats. These animals including cat, tiger, lion, cheetah share many features in common including facial character and body postures. These animals have larger possibility to fool the CNN model because of the sharing of these features and we want to experiment whether CNN model is still effective in this kind of classification.

With the origin feline images from ImageNet, we need to generate the input images (blocked feline image) for our dataset. To establish our blocked dataset, we implemented two kinds of block on our origin images:

Blind Blocking: Generating blocking parts with certain size randomly in the origin images

Non-Blind Blocking: Generating the blocking part with certain size at the center of origin images

2.2 Data generation

In our project, we use the images from ImageNet as our training, validation and test data. Since ImageNet only provide us with the URLs of the images, the step before training our model is to generate dataset with uniformed structure. Our idea is to build a data structure that fit the loading function in Pytorch. Observing the data structure of CIFAR10 and MNIST dataset, the dataset is a dictionary with key "data", "label", "classname" and data for validation the set. So the main part of the dataset should be "data" and "label". In our dataset, we should have one more keyword "blurdata" as for each image we take blur step. To mimic the structure, there are the several steps we took:

1. Rescale the image uniformly to 64×64
2. Place a block in the middle of each original image (non-blind); place blocks in random position of each original image (blind)
3. Append the dictionary value with rescaled image, blocked rescaled image and label for each image pair.
4. Based on the image number of each class, divide data into three sets (train, validation, test).
5. Zip the three sets into packed document.
6. Similarly, we also pack the images after inpainting and deblurring through CNN into "Deblurred.gz" in this way.

2.3 Image Inpainting

In this part, we used a convolutional neural network to inpaint blocked images and then deblur the inpainted images. The architecture of the network is inspired from Michal's paper[1]. The network is composed of multiple layers combining convolution layers with ReLU. The input and output are both 3-channel RGB images. We trained our model by minimizing mean squared error on the validation dataset, the MSE loss can be represented as follow:

$Loss = \frac{1}{2|D|} \sum_{(x_i,y_i)\in D} ||F(y_i) - x_i||_2^2$ The optimization method we use is adam optimizer. Further, we assume that the image after deblurring recovered the blocked part. Thus, we can fit that part in the blocked image with the deblurred one. Making masked images in this way can gain smallest difference with original images.

2.4 Image Classification

For classification, we used CNN network architectures since they are designed to process the data in form of multidimensional array (2D Images). We choose VGG16 and AlexNet because they both performs very well in Image classification, and we want to compare their performance on our inpainting Feline dataset.

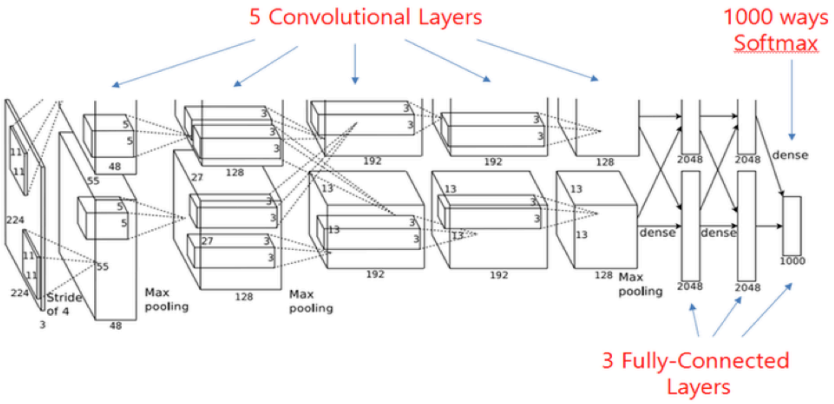

Figure 1: Alexnet Structure.

AlexNet contains eight layers including five convolutional layers and three fully connected layers.

VGG16 contained 13 convolutional layers with ReLu and Max-pooling between every convolutional layers and 3 fully connected layers.

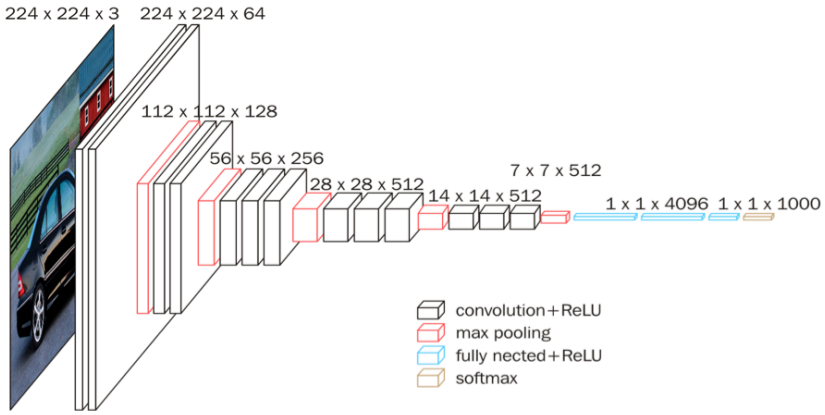

Figure 2: VGG16 Structure.

3 Experiments

3.1 Data generation

We read every image in raw image path and take steps as above. here are several examples of the original images and the blocked images.

Figure 3: Example of image blocking

- **Blind Inpainting**: For this approach we added two 8x8 masks at random locations in the data set, as can shown in Figure 3, which became our input to the network. Ground truth in this case were the true images themselves.
- **Non-Blind Inpainting**: One spatially fixed 16x16 mask was used to generate the corrupted input data set. Ground truth in this case was the true images themselves, as shown in Figure 3.

3.2 Image Inpainting

In inpainting step, we trained our inpainting model on blind dataset and non-blind dataset separately. These two datasets perform differently in inpainting After inputing the blocked image dataset to our CNN, we got the following results:

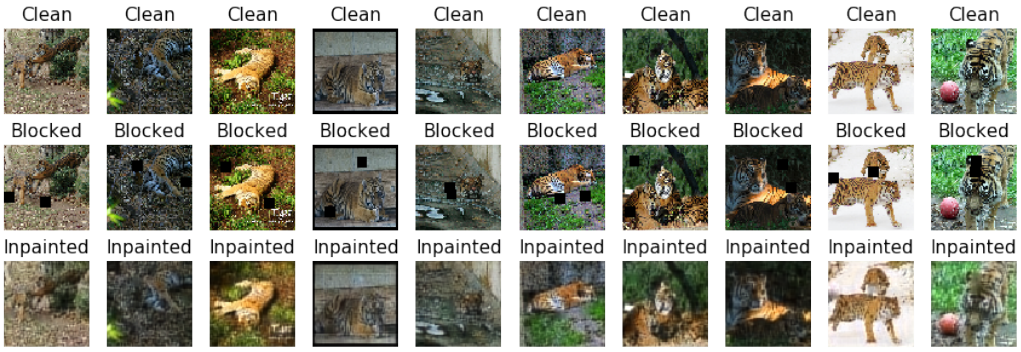

Figure 4: Blind Inpainting images.

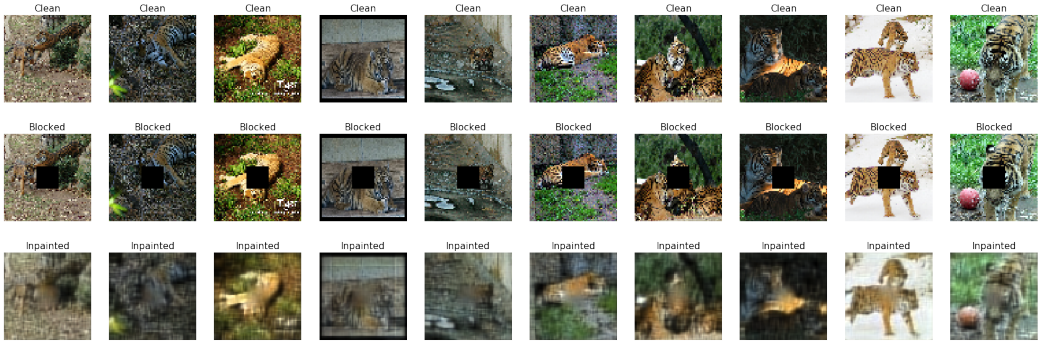

Figure 5: None-Blind Inpainting images.

Figure 4 indicating that the blind painting did not perform well and experiments on this were stopped after we are not able to achieve better result. Figure 5 shows that non-blind inpainting has a better performance than blind one since it was more likely to predict correct tiger features especially in the second pair from the right side. Obviously, the predicted images are kind of blurry so we decided to deblur those images base on CNN.

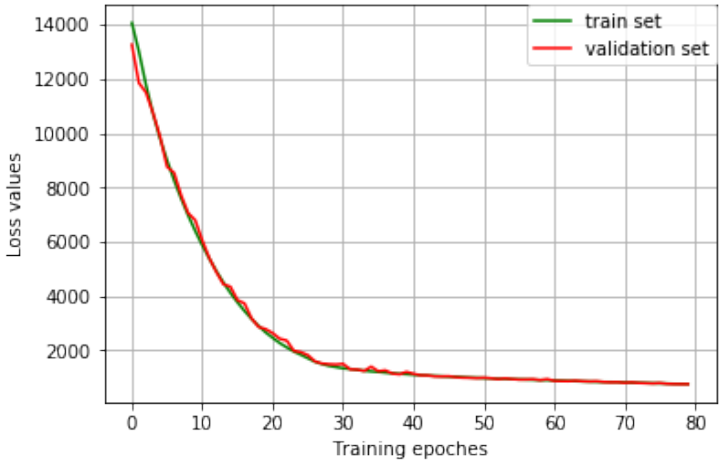

Figure 6: Training and test Loss.

For the inpainting CNN, there is an aggressive drop in the loss at beginning which is the result of the network greatly improving its performance over the random initialization. The tail end of the loss curves seem to flatten out, but the network is in fact still learning.

3.3 Image Deblurring

In this part, we are basically using the same CNN network structure for deblurring. In this case, the training data would be the recovered images which is blurry from last step. Here is the output of the network:

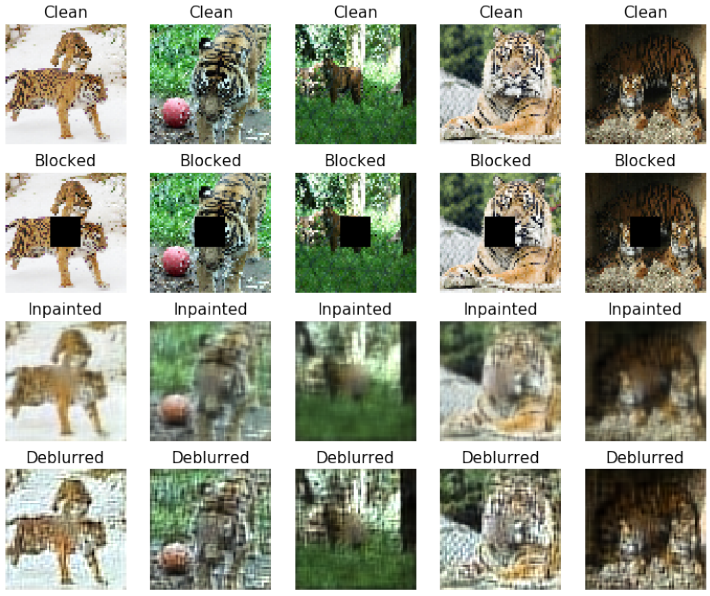

Figure 7: deblurred images.

Here we can find that the deblurred images in the last row are much more clean compare to inpainted images and more detailed features were added. Those indicates that our CNN works perfectly on deblurring blurry images.

3.4 Masking

In masking, we just fitted the blocked part in the blocked image with the same part in the deblurred image. Here is an example:

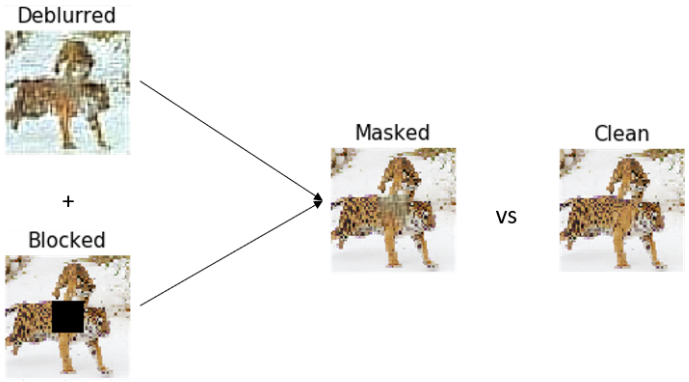

Figure 8: masking process

3.5 Comparison of different classification network

We resize the images into 224x224 and input them into Alexnet and VGG 16. We explored different methods (different optimizers and learning rates, weight decay and etc.) to examine how they affect these two networks performance. The following table is the hyper-parameter that have the best performance.

Table 1: hyper-parameter for Alexnet and VGG16 network

Network	Optimizer	Learning rate	Dropout	Batch Size	Weight Decay
VGG16	*Adagrad*	0.001	0.5	32	0.0005
Alexnet	*SGD*	0.001	0.5	32	0

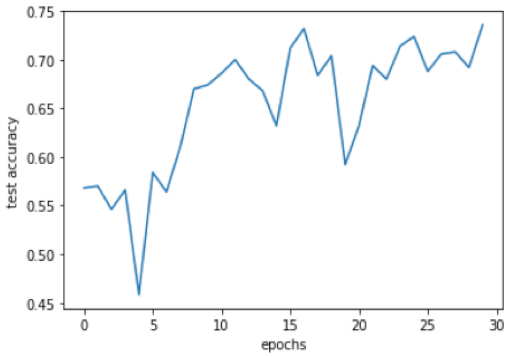

Figure 9: Alexnet test accuracy.

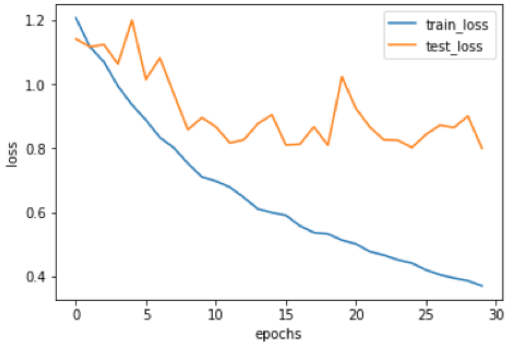

Figure 10: Alexnet test loss.

From the classification results of Alexnet, we are able to find that the loss is increasing slightly while the accuracy is fluctuating wildly even though we have random initialization. Those two facts indicates that the model is overfitting.

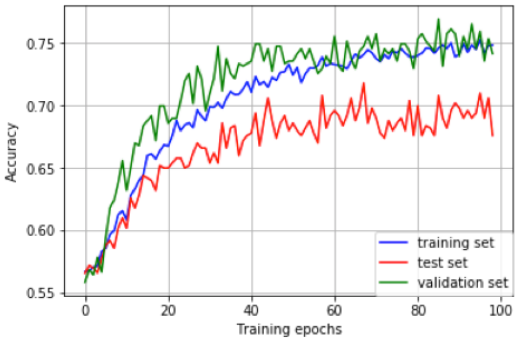

Figure 11: VGG16 test accuracy.

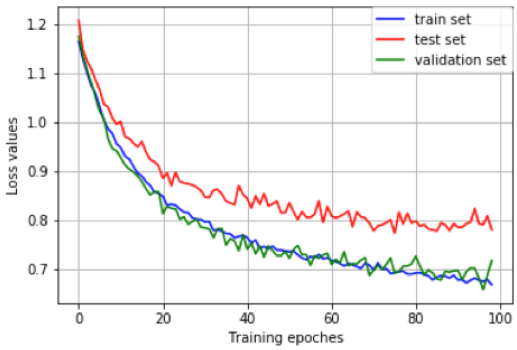

Figure 12: VGG16 test loss.

The vibrating accuracy basically means that some portion of examples is classified randomly, as the number of correct random guesses always fluctuate. In overfitting, a statistical model describes random error or noise instead of the underlying relationship. Thus, sensitivity to noise is a common definition of overfitting. Moreover, another evidence of overfitting is that the loss starts to increase. Loss is measured more precisely, and it is more sensitive to the noisy prediction if it's not squashed by sigmoid. Intuitively, we can imagine a situation when network is too sure about output (when it's wrong), so it gives a value far away from threshold in case of random misclassification.

The reasons causing overfitting may be:

1. Not enough training data points
2. The ordering of data points
3. Learning rate is too large

However, we have already shuffled the training data before inputting into the model, changed to small learning rates and applying batch normalization, but the results do not get much better. Finally, we stopped the Alexnet at 30th epoch (early-stopping) to prevent from overfitting heavily. In the future, we may need to collect more image data to overcome the overfitting problem on Alexnet.

VGG16, on the contrary, performs relatively much better. After training it for 100 epochs, the accuracy become less fluctuating and the loss finally converges to 0.79. Overall, it has a higher test accuracy and does not have overfitting problem. Thus, we use VGG16 instead of Alexnet to do the classification on the deblurred image data.

Table 2: VGG16 classification on different Test set

Classification Network	**Original accuracy**	**Blocked accuracy**	**Inpainted accuracy**	**Deblurred accuracy**	**Masked accuracy**
VGG16	60.00	50.60	53.20	61.00	59.40

From Table 2, we are able to find that the VGG16 model we used are not so power for feline classifying based on our limited dataset (only 60% on original image data). However, from the other four accuracy from every step we took, we found that the non-blind block at the center of the original images will significantly influence the classification of our CNN model. From the virtual result we know that the inpainting with blocked image actually recover the blocked part efficiently. When observing the classifying step, the inpainting not help classification a lot. Same with our assumption, the images after inpainting are like uniformly blurred image, so after we implemented a deblurring CNN model on those images, we improve the accuracy of classification a lot (61%). However, since the VGG model was not so well trained, we had the assumption that if the set have smallest difference

in accuracy with the accuracy of original dataset, the dataset best recover the blocked part. With this idea, the masked dataset actually has the accuracy nearest to original one.

4 Conclusion and Future Work

In this project we used a convolutional neural network for blind as well as non-blind image inpainting and deblurring. We found blind image inpainting to be a much harder problem than non-blind painting due to its uncertainty. To improve the model, we have several ideas:

For blind case:

1. Add features like the start points of blocking when we implementing blocks with fixed size.
2. Enlarge the dataset scale. When we have larger dataset, we can have less likelihood to fast overfitting.
3. Try on different blocking on the same image. For limit dataset case, this solution may help fitting more case of blocking. But may not so efficiently in preventing overfitting.

For non-blind case:

1. Since we know the part being blocked, we may just change our CNN model to have the output of our model just have the size of the blocked part. So, we can just recover the blocked part rather than a blurred whole image.
2. Similarly, enlarge the dataset scale. When we have larger dataset, we can have less likelihood to fast overfitting.
3. Since we have a quite large block at the center of each image, we may consider increase the complexity of our network to have higher level features that consider wider range in one image.

Furthermore, we tried out two different network architectures for image classification: Alexnet and VGG16. Because of the poor performance of Alexnet, we used VGG16 to do the classification. Unfortunately, the test accuracy we achieved is 60% which is a little bit lower than what we expected since the training accuracy is above 70%. The relative low accuracy rate indicates that VGG16 may not be so powerful to classify feline. In the future, we would like to consider using VGG19 or ResNet as the classifier. Overall, in this project, we gained practical experience with the training and fine tuning of convolutional neural networks and managed to produce reasonably good results.

5 Acknowledgement

We would like to thank our class teaching assistant Sainan Liu for providing technical supporting and answering questions.

We would like to thank Axium for his baseline code https://github.com/axium/CNN-For-End-to-End-Deblurring–Keras/blob/master/CelebA%20Deblurring.ipynb

6 References

[1] Hradiš et al.: Convolutional Neural Networks for Direct Text Deblurring. BMVC 2015.

[2] Gupta. Kushagr& Suleman Kazi. & Terry Kang.:DeepPaint: A Tool for Image Inpainting.