Image Orientation Detection

Tao Chen, Ruijing Bao  
Department of Electrical and Computer Engineering  
Texas A&MCollege Station, TX 77840

*Abstract*—In this project, we will present an application for detecting image orientation, using VGG-16 network based on CNN algorithm to extract features, and fix the orientation accordingly.

Keywords—image orientation, VGG-16, Guided Grad-Cam, Guided Backpropagation

# Introduction

Nowadays, digital cameras have come very common in our life. Every phone, laptop or tablet is equipped with one. Digital images are more common and easily produced than ever. One can take a photo in any place and time. After uploading the images to the PC, We notice that some of the images are rotated, and need to go through your album manually preform the fixes. The process is a time consuming and therefore nobody is doing so and the album staying as is unfixed.

In this project, we discuss the problem of rotating an image to its correct orientation. Initially, we have a data set of preprocessed image rotated by 0°, 90°, 180° or 270°. Then we train a deep convolutional neural network (CNN) to detect the correct orientation of the images. The training dataset comes from Indoor CVPR and Google Street View. Also, we apply Guided Grad-Cam and Guided Backpropagation to make the neural network visible and explain how the CNN makes the correction.

# CNN Algorithm & VGG-16

## CNN algorithm to image orientation detection

 For image orientation detection, some software tools have the capability to straighten them, provided that the image is already in the right orientation. To illustrate that, let’s take a look at the *Level* tool of *Adobe Photoshop Lightroom*. As you can see in the images below, this kind of tool can correct small rotations with high accuracy (look at the edge of the road as a reference for the horizon.



Figure 1:Left: Original image. Right: Image corrected with Lightroom's Level tool.[7]

However, it is very limit to a very specific image dataset because the features of the images are variable and hard to classify them. For example, in the previous case, we would easily figure out that the image is upside down by acknowledging the position of the sky and the road. But in another case like a person, it is not the case.

Hence, it’s where CNN comes in. **Convolutional neural networks** (CNNs) are good at processing data that can be spatially arranged (2D or 3D). Typical use cases of CNNs are object detection and recognition. And based on the CNN algorithm, it will be very easy to get the features of all possible scenarios.

CNNs are organized into interconnected layers of artificial neurons. The first layer’s input is connected to the raw data that we want to process (images, text, etc.) and the last layer output is whatever we want to predict.

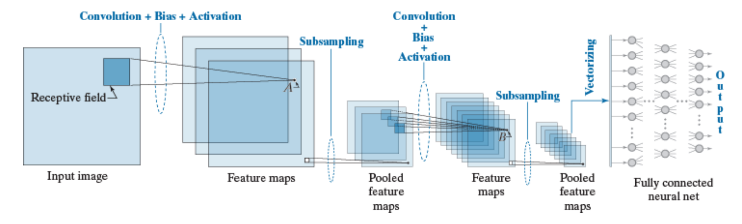


Figure 2: CNN Architecture [4]

The purpose of the transformations that take place at each layer is to compute **features**. In machine learning, features are attributes that simplify the representation of the data. While traditional feature extraction methods are manually implemented by humans, the CNN algorithm can extract the optimal feature by itself. And in general, when the networks are trained with more data, it will learn higher level features based on the dataset.

Therefore the networks will learn its own features to make the best decision. The only thing we need to do is feeding the labelled data to the networks. When we feed the data, the networks will train these data and update the weights of every layer continuously. In other words, as the training progresses, the weights are updated to produce better features which help the network make better predictions.

More specifically, there are three most important layers in the CNN.

**Fully-connected layer**: this is the simplest type of layer. It connects all its inputs to all the outputs of the preceding layer. It is usually used for making decision.

**Convolutional layer**: this is the type of layer that performs most of the computation in a convolutional neural network, hence their name. In essence, convolutional layers operate in a similar way to fully-connected layers. The difference is that the neurons are small kernels only connected to a small portion of the inputs, as opposed to all of them. These kernels are small two-dimensional windows that are slid over all the spatial locations in an image in order to compute a specific feature.

**Pooling layer**: this type of layer down-sample its input. Similar to convolutional layers, pooling layers consist of small sliding kernels that simply average spatial regions (average pooling) or take the maximum value (max pooling). Down-sampling is typically used in convolutional neural networks to reduce the number of weights in consecutive layers, which in turn reduces their computational complexity.

What is more important is that unlike to Convolution layers, the pooling layers usually extract the main features of the all features that extracted by the Convolution layer hence reduce the computation.

And there are some abstract introduction of the CNN architecture and the main idea of the algorithm.

## VGG-16 NET

Last paragraph we introduce the main content of the CNN algorithm. And nowadays, the common technique for building a CNN classiﬁer for a new domain is to adopt an architecture originally designed for the ImageNet dataset, modify it, and apply it to the new domain which used the conception of Transfer learning.

Then we found that VGG-16 Net is a great architecture for our dataset to transfer since it is very expandable and generalizable to other image datasets. And the way to use the transfer learning is that we keep the main parameters of the VGG-16 model and just replace the 1,000 outputs corresponding to the 1,000 objects with 4 outputs corresponding to 0◦, 90◦, 180◦, or 270◦. [1] And the parameters mean that the weights of the whole networks. The only layer we need to train with is the fully-connected layer. This indicates that we are doing some transfer learning: VGG-16 detects 1000 classes of objects, which would be useful for detecting orientation. Likely initializing our weights to those of VGG-16 makes our network converge to nearby values of the weights.

Then we introduce the VGG-16 network briefly. VGG-16 is developed by Oxford university and Google which is a deep convolution neural network. It constructs the simple architecture by repeatedly add the 3\*3 convolution kernels and 2\*2 pooling layers over the whole neural networks [6]. It is the structure of the VGG-16.

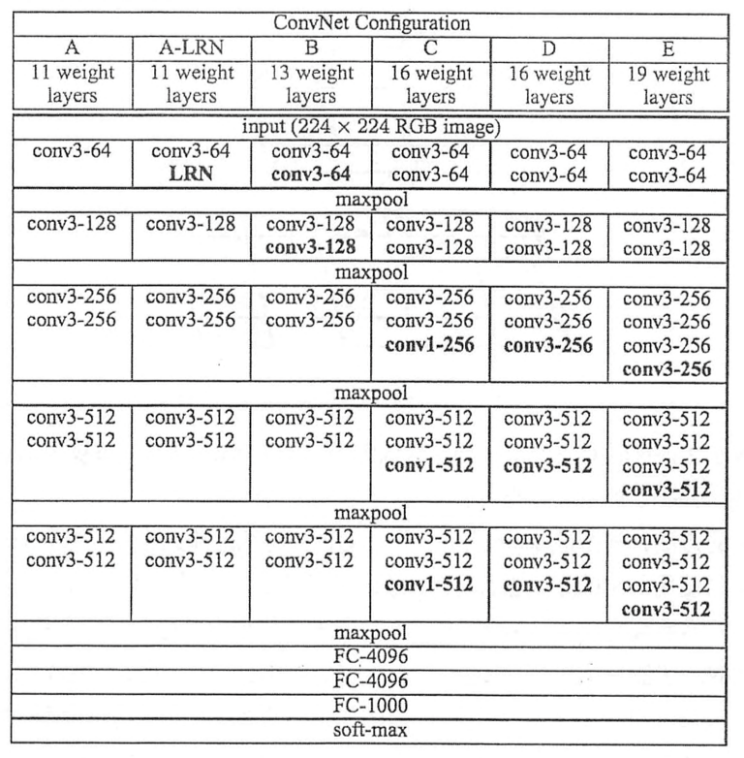


Figure 3: VGG-16 Structure[6]

We can know that the VGG-16 has three convolution layers in a block of one convolution and in every convolution, it will have a max pooling layer to reduce the complexity of the features. And in the VGG-16, there are many 3\*3 convolution layers connected which is very useful in parameter reduction.

Hence, likely initializing our weights to those of VGG-16 makes our network converge to nearby values of the weights. The set of photos is transformed by rotating all the photos in the original training set by 0◦, 90◦, 180◦, or 270◦. The VGG architecture requires that the input be of size 224 × 224 × 3. We resize the input image to ﬁt inside a 224 × 224 square, and pad it as necessary with black pixels in order for the input to be 224 × 224.

# Exeperimental Results

## Dataset Description

Google Street View dataset contains 62,058 high quality Google Street View images with a total number of 10343 placemarks. The images show street scenes of urban and rural areas in different cities, so the dataset contain most outdoor scenes we can see in our daily life. By using this dataset, we train the VGG-16 to detect orientation of outdoor scenes.

Most scene recognition models have good performance in outdoor scenes but poor in indoor scenes. So we train the other dataset, the Indoor CVPR dataset, to detect orientation of indoor scenes. The dataset contains 67 indoor categories and 15620 images. The scenes provided by this dataset are full of both local and global discriminative information, which means more details of scenes will be contained in the dataset. For example, the corridor can be recognized by its global spatial feature, which is a most significant one; however the bookstore has to be recognized by the objects it contains (e.g. books).

By using the two datasets mentioned above, we believe that the VGG-16 model can detect the orientation for both indoor and outdoor images, and it can meet our daily demand for correct the orientation of photos to the correct one.

We split all datasets into training (75%) and test (25%) sets, and then transform each of the sets by randomly adding one of the four rotations in each photo.

## Experimental Results

Before we implementing the VGG-16 model, we randomly choose the images in Indoor CVPR dataset, and rotate the original images by 0°, 90°, 180° and 270° respectively. The purpose for preprocessing the images is to create a new dataset with various rotated angles, so that we can implement the VGG-16 to detect the correct orientation.

We classify the preprocessed image into 4 groups (i.e. 0°, 90°, 180° or 270°), and calculate their accuracy separately. Since the preprocessed image has been put into four folders named by their rotated angles respectively, we compare the corrected angle obtained by VGG-16 with the rotated angle. The accuracy rates for the two datasets are shown in *TABLE I* and *TABLE II*.

1. *Orientation detection accuracy for indoor images*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rotated Angle | 0° | 90° | 180° | 270° |
| Accuracy | 93.8% | 92.9% | 91.8% | 92.4% |

1. *Orientation detection accuracy for Srteet View images*

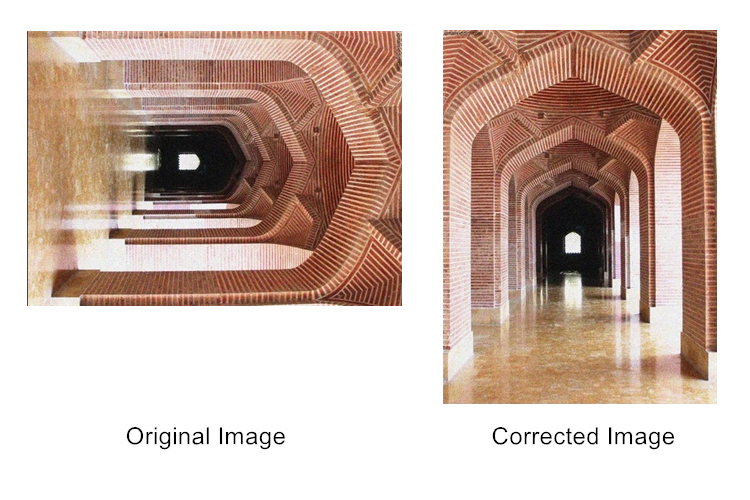
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rotated Angle | 0° | 90° | 180° | 270° |
| Accuracy | 99.8% | 99.4% | 99.7% | 99.6% |

## Discussion

As we can see from the table, the accuracy for all rotated angles is above 90%. And the VGG-16 model have the highest accuracy correcting angles if we do not rotate the angle of images, as we expected. As described in the previous section, the VGG-16 model work well for detecting orientation of outdoor pictures, but the accuracy is a little lower when detecting orientation of indoor pictures due to the complex situation. No matter which category the rotated images belongs to, the accuracy rate is satisfying.

One thing we need to mention is that we also tried to use our own-collected pictures to test the model. We found that it has good performance in most pictures, but if the resolution of the image is low, the result is unsatisfying. This experimental result is also mentioned in [1].

## Some orientation detection results



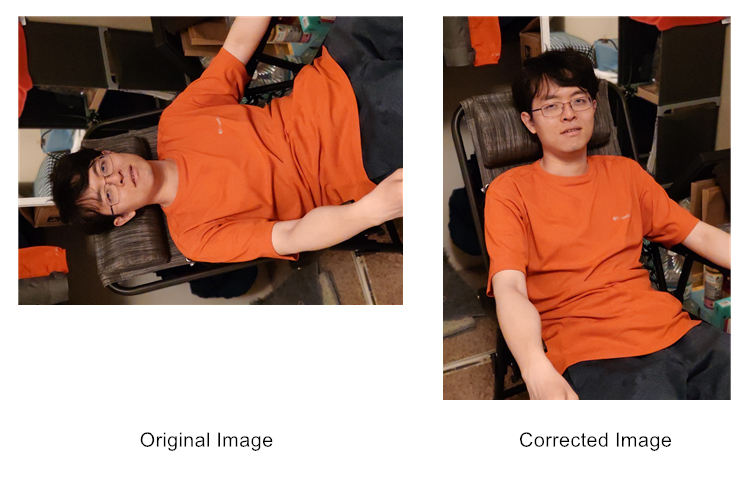
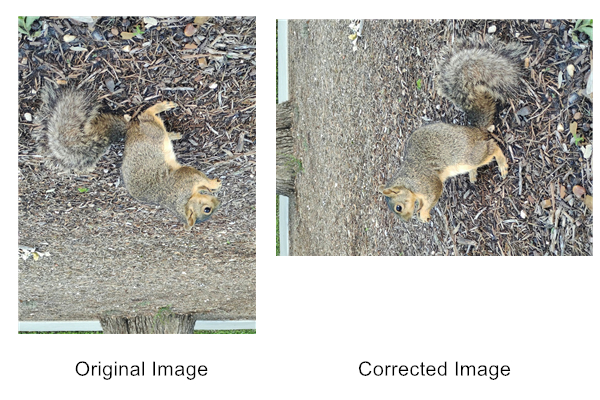


Figure 4. Orientation correction for images with different rotated angles

In Figure 4 we show some indoor and outdoor image orientation correction results. Orientation of three of the images are detected correctly, but the squirrel picture is rotated in a wrong angle. The reason why the VGG-16 makes such decisions to correct image orientation will be explained in the next section.

# Guided Backpropagation And Grad Cam

## Guided Backpropagation

Basically, we get the conception of the CNN algorithm and the VGG-16 model in Section II. However, we still don’t know why it should be right and trustworthy because the neural network is a black box progress. Hence, we don’t truly know what happened exactly.

We use a variant of Guided Backpropagation to visualize important features for the machine to recognize the image. [2] Guided Backpropagation computes a modiﬁed version of the gradient of a particular neuron with respect to the input. It is the reverse convolution progress, which is based on the conception of the backpropagation. In other word, it is the derivate of the input.

The idea of the Guided Backpropagation is that neurons act like detectors of particular image features. Because we only care about the features that the neuron detects, we can set that all the negative gradients to 0. We only back propagate the gradients that the input and the gradient are all lager than 0 which is the mean of Guided.

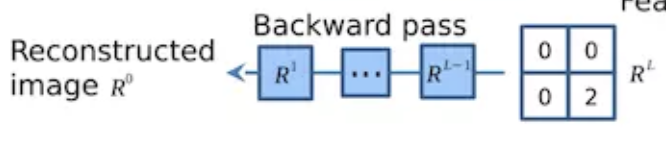


Figure 5. Set to zero all activations except one [2]

Guided backpropagation is a way of visualizing which pixels provide evidence for the presence of features in the input image that inﬂuence the output neuron [2]. The pixels that are visualized never depress features causing the neuron of interest to activate. Instead, they only activate features throughout the layers of the network. However, because of it, there is also a problem. All the features that it extracts will be shown when the gradient is larger than 0. Hence, we cannot figure out which is the most important feature to decide the result, as shown in figure 6.



Figure 6. Guided Backpropagation

We can see that the intensity of the image is different, but can we find a way to make it clearer?

## Grad Cam

In the last paragraph, we discuss a way to make the feature visualize. Here we implement a way to find the most important decided feature which is named Grad Cam.

It is very similar to the Guided Backpropagation algorithm back propagating the gradients of every layer’s input and deconvolution the network to get every feature.

However, the Grad Cam algorithm gets every feature’s weight and then sum up the weighted sum. And it uses the global average of the gradients to calculate the weights [3]. If we define the kth feature map’s weight to be , then

where Z is the number of pixels in the kth feature map, is the value of input before the softmax layer, is the pixel value of the point (i, j) in kth feature map.

Figure 7 shows the structure of the Grad Cam.

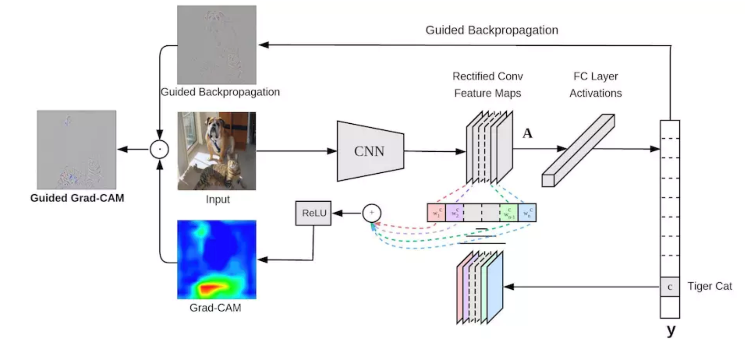


Figure 7. Grad Cam Structure [3]

A ReLU function will be added to the weighted sum as shown in Figure7. We only care about the pixels that have positive influence on the classification. If we do not add a ReLU function, it may bring in some pixels that are not included in this classification.

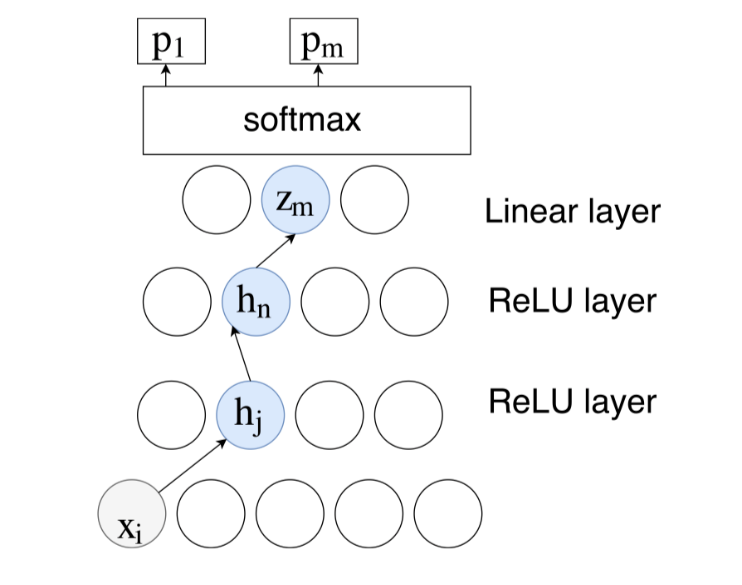


Figure 4: ReLu function in the Grad Cam [1]

## Guided Grad-Cam

We combine Guided Backpropagation and Grad-Cam together to show the most significant features. The result of the Guided Grad-Cam for figure 6 is shown in figure 7.

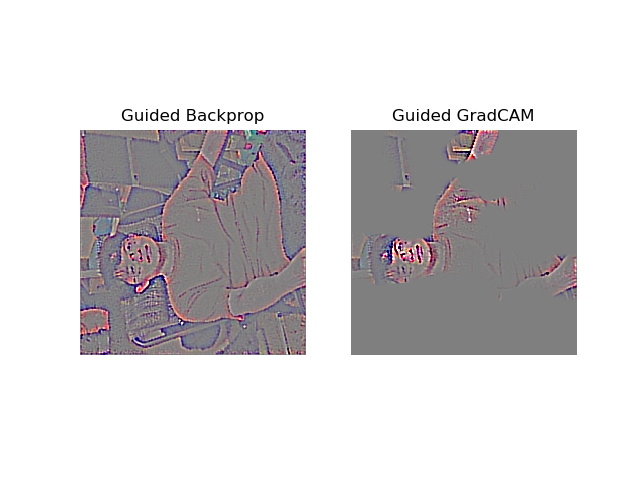


Figure 7. Guided Grad-Gram Result of Man

We can conclude from the result that the man’s head and arms are the most significant features in this picture. That explains why VGG-16 rotated the image 270° counter-clockwise to get the correct orientation.

Another Guided Gram-Cam result we obtained is shown in Figure 8.

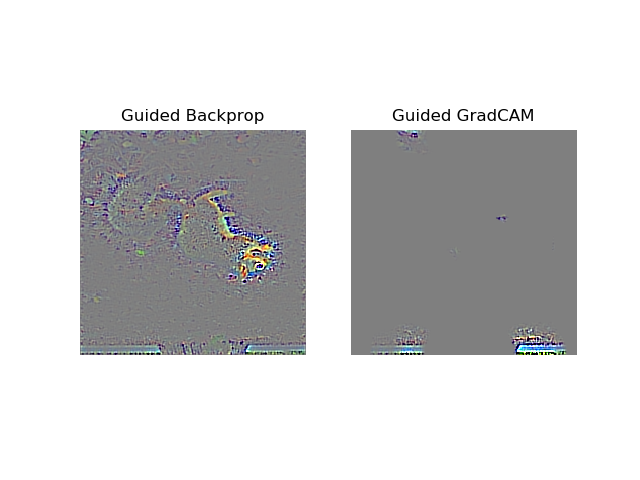


Figure 8. Guided Grad-Gram Result of Squirrel

These two pictures show Guided Backpropagation and Guided Grad-Cam of Squirrel. From the Guided Backpropagation result, we found that boundary of the squirrel is mixed with the background. This is because the color of the squirrel is similar to the ground. In the Guided Grad-Cam, the VGG-16 model takes edges of the road to be the most significant feature, but it may label the edge of road to be the edge of buildings, so the image is rotated 270° counter-clockwise, which is a wrong result.

# Conclusion

In this report, we demonstrated that a deep convolutional neural network, specifically the VGG-16, has good performance in image orientation detection. We used Guided Grad-Cam to visualize how the VGG-16 correct the orientation of an image and explain both the correct and wrong result through visualization. Since our rotation of images is limited to four specific angles, correcting the images in all rotated angles can be accomplished in the further work of image orientation detection.

##### References

1. U. Josh, and M. Guerzhoy. "Automatic Photo Orientation Detection with Convolutional Neural Networks." 2017 14th Conference on Computer and Robot Vision (CRV). IEEE, 2017.
2. J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, “Striving for simplicity: the all convolutional net,” in 2015 3rd International Conference on Learning Representations, 2015.
3. Y. Selvaraju R R , Das A , Vedantam R , et al. Grad-CAM: Why did you say that? Visual Explanations from Deep Networks via Gradient-based Localization[J]. 2016.
4. Zixiang Xiong ECEN 642 Charpter 13.
5. K. Simonyan, and Z. Andrew. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).
6. Daniel Saze. Correcting image orientation using convolutional neural networks, https://d4nst.github.io/2017/01/12/image-orientation/