Machine Learning Engineer Nanodegree Capstone Proposal

Xing Ye
29th January 2018

Domain Background

Visual object recognition entails much more than determining whether the image contains instances of certain object categories. Convolutional neural networks (ConvNet) is one of the three styles of the object recognition algorithms [8]. In machine learning, a convolutional neural network (CNN) is a class of deep feed-forward artificial neural networks that have successfully been applied to analyzing visual imagery [3]. ConvNet apply a series of filters to the raw pixel data of an image to extract and learn higher-level features, which the model can then use for classification.

For image recognition, convolutional neural networks are the current state-of-the-art model architecture. Recent work shows ConvNet have remarkable ability to localize objects even without object annotation [8].

Problem Statement

Conventional convolutional neural networks have a brilliant ability on processing visual task since the lower convolutional layers extract features of various depths from the input images. However, the excellent talent is lost when fully-connected layers used for classification. Additionally, the fully-connected layers are prone to overfitting that hamper generalization of the overall network [2]. Recently, some popular fully-convolutional neural networks have been proposed to avoid the use of fully-connected layers to minimize the number of parameters while maintaining high performance [1].

Despite the ConvNet can achieve high accuracy in image recognition, we have no idea which areas are important for the model to predict. Learning deep features for localization is critical to better understand ConvNet.

Datasets and Inputs

The project will use the Dogs vs Cats Dataset[9] which released by the competition hosted on Kaggle in 2014. In this project, we use the train file and a sample of test file. In this project, the training archive contains various sizes of 25,000 images of dogs and cats. There are 12.5K images of dogs and the same number of cats.

Before inputting the model, we resize the original images into the same size of 128x128x3 (128 are the height and width of the image) by interpolation. The number of channels is 3 since our inputting image is RGB. Each image in the training file is named for its category and number.

Solution Statement

Problem is described in the problem statement. Dropout is proposed by Hinton *et al.* [5] as a regularizer which randomly sets half of the activations to the fully connected layers to zero during training. It has improved the generalization ability and largely prevents overfitting [4].

Another strategy proposed by Lin et al.[2] replaces the traditional fully-connected layers with global average pooling layers (GAP). GAP is more native to the convolution structure by enforcing correspondences between feature maps and categories. Additionally, there is no parameters to optimize in the global average pooling layer thus overfitting is avoid in this layer. Recent work by Zhou *et al*[1] has shown that the global average pooling layer explicitly enables the convolutional neural network having remarkable localization ability.

In the project design, We will explain the procedure for generating the class activation maps (CAM) using average pooling layer (GAP) in CNNs.

Benchmark Model

ConvNet contain three major components: Convolutional layers, Pooling layers, and Dense (fully connected) layers. Convolutional layer performs a set of mathematical convolution to produce values in the output feature map and it typically applies a RELU activation function to the output to introduce nonlinearities into the model. The pooling layer down-samples the image data extracted by the convolutional layer to reduce the dimensionality of the feature map so as to decrease processing time. Dense layers perform classification on the features extracted by the convolutional layers and down-sampled by the pooling layers [6].

We will build a simple ConvNet and replace the last max pooling layer with average pooling layer to generate one feature map for each corresponding category. We add one dense layer at the top layer to observe the performance in the training period.

Evaluation Metrics

Accuracy

Accuracy is a common metric for binary classifiers and it equally considers both true positive (TP) and true negatives (TN).

$$\text{Accuracy} = \frac{TP + TN}{dataset\ size}$$

Loss

We use loss value to evaluate the performance of our model during the training phase. For a single example in the training set, we optimize the weighted binary cross entropy loss

$$L(X, y) = -w_{+}*y*\log(Y=1|X) - w_{-}*(1-y)*\log(Y=0|X)$$

Where $p(Y=i|X)$ is the probability that the network assigns to the label i, $w_{+} = |N|/(|P| + |N|)$, and $w_{-}=|P|/(|P| + |N|)$ with $|P|$ and $|N|$ the number of positive cases and negative cases of dogs in the training set respectively.

Project Design

In this work, we will build a 18-layer convolutional neural network trained on the Dogs vs. Cats dataset. We replace the final max pooling layer with average pooling layer for generating the class activation mapping and add one dense layer that has a single output, after which we apply a sigmoid nonlinearity [3]. We use dense connections and batch normalization to make the optimization of our ConvNet tractable. The architecture of our simple has three components.

The first part is three convolutional block that performs as a feature extractor. Each block is consist of one convolutional layer and a max pooling layer. The input to the first convolutional layer is our image data with the identical size. The next to every convolutional layer is batch normalization thus we can ignore the initialization of the previous layer weight.

The second part is a convolutional block which is composed of one convolutional layer followed by an average pooling layer. The average pooling layer performs dimensionality reduction, where a tensor with dimensions $(\mathrm{h}, \mathrm{w}, \mathrm{d})$ is reduced in size to have dimensions $(1,1,\mathrm{d})$ by averaging all h*w value. The average pooling layer is used for generating class activation maps.

The third part is a densely (fully-connected) layer with a activation of sigmoid. The output from the last convolutional block is vectorized and fed into this densely layer. The output is 1 if the inputting is a dog image.

We save our fully trained model after training the simple model on our dataset. The next is generating the class activation map when loading a new dog/cat image from the test file into our trained model.

The following is the main procedure to generate the class activation map.

```
def new_model_and_weights(model_path):
    # get the model
    model = load_model(model_path)
    # get AMP layers weights
    all_amp_layer_weights = model.layers[-1].get_weights()[0]
    # extract wanted output
    new_model = Model(inputs=model.input,
                      outputs=(model.layers[-4].output, model.layers[-1].output))

    return new_model, all_amp_layer_weights


def model_CAM(img_path, model, all_amp_layer_weights):
    # get filtered images from convolutional output and model prediction vector
    image = pretained_path_to_tensor(img_path)
    # change dimensions of last convolutional output to 12 x 12 x 128
    last_conv_output, pred_vect = model.predict(image)
    # change dimensions of last conv output to 12*12*128
    last_conv_output = np.squeeze(last_conv_output)
    # get model's prediction (number between 0 and 1, inclusive)
    pred = np.argmax(pred_vect)
    # get AMP layer weights
    amp_layer_weights = all_amp_layer_weights[:, pred]
    # bilinear upsampling to resize each filtered image to size of original image
    mat_for_mult = ndimage.zoom(last_conv_output, (128/12, 128/12, 1), order=1)
    x = mat_for_mult.reshape((128*128, 128))
    # get class activation map for object class that is predicted to be in the image
    final_output = np.dot(x, amp_layer_weights).reshape(128,128)
    # return class activation map and prediction
    return final_output, pred
```

In the function of *new_model_and_weights*, we extract the activation maps for all classes and redefined the output of the model. The function of *model_CAM* generates the class activation map of the category the trained model predicted. The dimensions of the data need to transform to meet the resized inputting image. The output of this *get_cam* file is an added image between the original image and the class activation map.

Reference

[1] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, Antonio Torralba, Computer Science and Artificial Intelligence Laboratory, MIT. Learning Deep Features for Discriminative Localization, 2016.

[2] Min Lin[1,2], Qiang Chen[2], Shuicheng Yan[2]. Network In Network.

[3]https://en.wikipedia.org/wiki/Convolutional_neural_network

[4] Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems 25, pages 1106–1114, 2012.

[5] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580, 2012.

[6]https://www.tensorflow.org/tutorials/layers

[7]M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. Proc. ECCV, 2014. 2, 3

[8]Maxime Oquab*, Leon Bottou †, Ivan Laptev*, Josef Sivic*. Is object localization for free? – Weakly-supervised learning with convolutional neural networks

[9]https://www.kaggle.com/c/dogs-vs-cats/data