DeVISE: A Deep Visual-Semantic Embedding Model

Tsung-Yuan Tseng, Sin-Fu Huang

Abstract

Nowadays, in image classification models, it is difficult to process large numbers of object categories. With the growth of that number, collecting enough training data will cost a lot. In our milestone, we implement a paper, deep visual-semantic embedding model(DeViSE), which is published in 2013. The model in the paper is trained to identify visual objects using both labeled image data as well as semantic information gleaned from the unannotated text. Besides, we try some methods to improve the models performance. In the pre-training visual model, instead of AlexNet, we try different models and get the best accuracy on ResNet-110. In the pre-training language model, we use Googles pre-trained Word2Vec model and compare the performance between different text embedding dimension. In the DeViSE model, we got the image embedding from different layers output of ResNet and we found the layer before the global average pooling has the best accuracy in the DeViSE model. In zero-shot learning, we try to predict the image label of the dataset which different from the pre-training visual model we use but only have good performance in some classes.

1. Introduction

In DeViSE, they present a deep visual-semantic embedding model used to identify visual objects using both labeled image data and semantic information gleaned from the unannotated text. They demonstrate that this model matches state-of-the-art performance on the 1000-class ImageNet object recognition challenge while making more semantically reasonable errors, and also show that the semantic information can be exploited to make predictions about tens of thousands of image labels not observed during training. Semantic knowledge improves such zero-shot predictions achieving hit rates of up to 18% across thousands of novel labels never seen by the visual model.

In this paper, we go through in detail how we put Deep Visual-Semantic Embedding Model into practice, and the ideas we pop up to improve the model.

2. Baseline Model

2.1. Dataset for visual model pre-training

$Cifar$ 100 is a 100 classes image datasets containing 32x32 pixels for each image.

Link **:**
https://www.cs.toronto.edu/ kriz/cifar-100-python.tar.gz

2.2. How to import data

After importing TensorFlow, one can include data through a comment below.
`tf.keras.datasets.cifar100.load_data`

2.3. Additional library

Including $tensorflow$, $numpy$ and high-level API $tflearn$.

2.4. Other environment setting

Install $python3.6$, $jupyter$, $NvidiaDriver$, $Cuda8.0$ and $CuDNN6.0$,

2.5. Training process

2.5.1. Goal

We aim to implement the paper(Andrea Frome, 2013) baseline, that is, pretraining visual model and text model. After that, one can train a linear transformation to map from image vector to text embedding.

2.5.2. Network Structure

For the visual model, we used state-of-the-art ResNet model(Kaiming He, 2015). While we constructed 110 layers due to limited computation power, we get accuracy about 67 percents. For embedding space of text, we use Google News pre-trained skip gram model. In the next step, we retrieve the trained visual model image vector from the global average pooling layer. Finally, we trained a linear transformation map from the image vector to its label embedding.

$$\boldsymbol{Loss - Function:}$$

$$loss(image, label) = \sum_{j \neq label} max[0, margin - \vec{t}_{label} M\vec{v}(image) + \vec{t}_j M\vec{v}(image)] \quad (1)$$

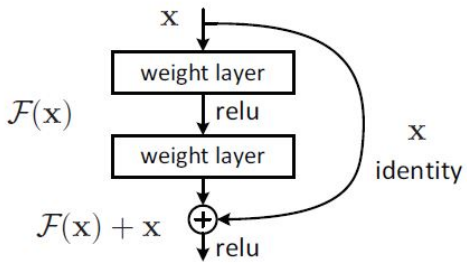

Figure 1. A building block from(Kaiming He, 2015). That paper result won the 1st place in the ImageNet localization task in ILSVRC 2015.

2.6. Baseline Result

After training the linear transformation, the accuracy roughly equals 43.7 percents. Since we retrieve image vector from global average pooling which is 64-D, we though that image embedding should be larger than 300-D(text embedding is 300-D). Therefore, the next step we will try to retrieve image vector from the activation layer which is the layer before the global average pooling to hope to get better accuracy.

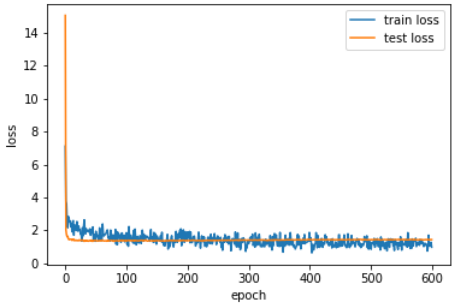

Figure 2. Training loss and testing loss.

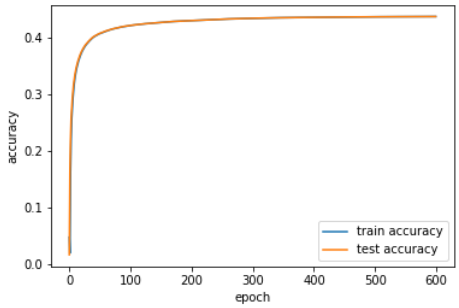

Figure 3. Training accuracy and testing accuracy.

Model type	Accuracy(%)
Softmax baseline	67.5
DeViSE	43.7

Table 1. Comparison of Deep Visual-Semantic Embedding Modal performance on CIFAR-100 data set.

Model type	Accuracy(%)
Conventional CNN -32 layers	32.1
All Convolution Net -32 layers	53.4
Residual Net -110 layers	67.5

Table 2. Comparison of pre-train visual model performance on CIFAR-100 data set.

3. Improved model

3.1. Describe Dataset

We use the same image dataset(Cifar100), but trying different pre-trained text embedding which is freebase-vectors-skipgram1000-en released by google.

Link **:**
https://github.com/3Top/word2vec-api

This data set is trained based on Google News and have 1000 number of dimensions. We firmly believe that the higher the text embedding is, the higher the accuracy is. The DeViSE model will increase roughly 1% to 2% accuracy as using 1000-D text embedding which makes sense.

3.2. Math formula

Our work has also tried loss function equal to cosine similarity plus euclidean distance. The reason is that if two vectors are closed, which means theta and the endpoint of the two vectors are as small as possible.

$$\boldsymbol{Loss - Function}:$$

$$loss(image, label) = \sum_{j \neq label} max[0, margin - \vec{t}_{label} M\vec{v}(image) + \vec{t}_j M\vec{v}(image)] + euclidean(\vec{t}_{label}, \vec{v}(image)) \quad (2)$$

However, as we implement the model using this loss function, the accuracy would drop by 9% to 10%. Thus, in the below sections, we implement the model using loss function as equation 1.

3.3. Network Structure

DeViSE paper mentions that one can retrieve the pre-train visual model's tensors for each image from the layer before the softmax layer which is the global max-pooling layer. As the previous section saying, we got roughly 43 percent accuracy. The problem is the mapping is from 64-D to 500-D(the dimension of text embedding), and the information of images is so severely compressed that the model can't learn the mapping well. Thus we pop up an idea that we retrieve the re-train visual model's tensors for each image from the layer before global max pooling which is the activation layer, and the dimension of this layer is 2048.

The result of this improved model accuracy significantly. The accuracy is roughly 64 percent compared to the baseline 43 percent accuracy. Besides, we have tried getting tensors from the layer before the activation layer which is a batch normalization layer, but the accuracy drops to roughly 32 percent.

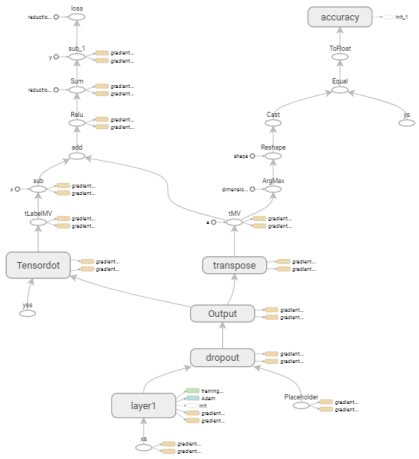

Figure 4. Visualized tensorflow from tensorboard.

3.4. Table for result

In table 4 we can find that our model's accuracy is higher than DeViSE baseline.

Model type	Accuracy(%)
DeViSE baseline(text embedding 300-D)	43.7
DeViSE baseline(text embedding 1000-D)	44.5

Table 3. Comparison of using different pre-trained text embedding.

Model type	Accuracy(%)
DeViSE baseline	43.7
activation layer-based DeViSE	64
batch normalization layer-based DeViSE	32

Table 4. Comparison of improved DeViSE baseline with DeViSE baseline.

4. Result

The figure5 and figure6 are the results of our improved DeViSE model. In the training process, we observe that the

model would suffer from overfitting. Therefore, we use the dropout technique so as to make it better, and it actually works better. In Figure6, it is important to note that one may observe the training accuracy is very high in the initial state. This is normal because the model we implement is to train a linear transformation map from visual embedding to text embedding, and the accuracy of the pre-trained residual net model is 92% and 65% for train accuracy and test accuracy respectively. Thus, the training accuracy for the DeViSE model in the initial state makes sense to some degree.

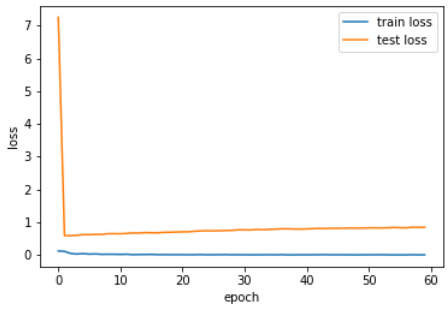

Figure 5. Training loss and testing loss.

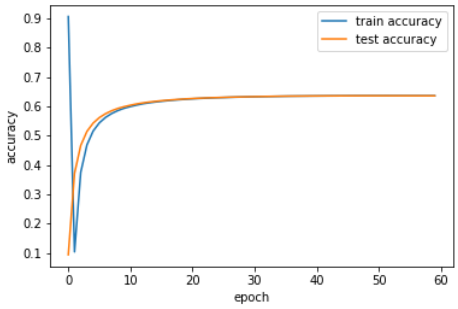

Figure 6. Training loss and testing loss.

5. Zero shot learning

After our improved DeViSE model training finished, we cache the parameters, and feed the unseen image data set which is cifar10, yielding an embedding. Next step we find the closest embedding vectors across all text embeddings. However, due to limited computation power. Instead, we using the following way to solve this issue. That is, we only find closest embedding vectors across cifar10's 10 labels embedding. The procedure is that if one of the Cifar10 image data feeds in the DeViSE model, and it would produce 10 classes multinomial distribution. The result is that cat label and $ship$ label get 17.53% and 14.67% respectively.

cifar10 label	Accuracy(%)
cat	17.53
ship	14.67

Table 5. Comparison of improved DeViSE baseline with DeViSE baseline.

6. Conclusion

To summary our work, we use the state-of-the-art residual net as the pre-trained visual model. Besides, we retrieve 3 different layers tensors from the residual net, and find that if one gets tensors from activation layers will get the best accuracy. In the pre-trained text embedding model, we have tried two text embedding which is 300-D and 1000-D respectively, and the latter would get a little higher accuracy. Finally, we feed unseen dataset cifar10 find that our improved model learn the better pattern in images of cat and ship.

In our future work, we think that the text embedding vector should give more information for the DeViSE model. If the close semantic embedding vector can be a little far away from each, the DeViSE model would gain more useful information. Therefore, to make a pre-train text model be more suitable is a must.

References

Andrea Frome, Greg S. Corrado, Jonathon Shlens Samy Bengio Jeffery Dean Marc' Aurelio Ranzato Tomas Mikolov. Devise:a deep visual-semantic embedding model. In *In advances in Neural Information Processing Systems*, pp. 2121–2129, Google, Inc. Mountain View, CA, USA, 2013.

Kaiming He, Xiangyu Zhang, Shaoqing Ren Jian Sun. Deep residual learning for image recognition. 2015.

Zhang, Ziming and Saligrama, Venkatesh. Zero-shot learning via semantic similarity embedding. 2015.