Project 1
CIS6930 Neural Networks and Deep Learning

Ashutosh Modi (modia), MSCS
Department of Computer Science and Engineering
University of South Florida
Tampa, Florida 33620
Email: modia@mail.usf.edu

***Abstract*—To check the performance of finetuning on residual nets for new dataset I modified last few layers in the pretrained ResNet50 model. Keras, an opensource python library, has been used to load a trained network. A residual network proposed by Microsoft research, called ResNet, was loaded from Keras and modified for new data. After detailed study it is observed that, even though it is easy to alter and add new layers, tuning hyperparameters to achieve good prediction accuracy requires large dataset and rigorous training.**

***Index Terms*—Transfer learning, CNN, ResNet, Hyperparameter tuning,**

1. Introduction

In this project, attempt is to classify and recognize different states of the food item. Dataset of food items like bread, tomato, butter, cheese etc. are collected in different states. These states are like cream, juice, diced, grated etc. Objective was to use pretrained weights for popular deep learning models and finetune them for our dataset. Open source library Keras has been used to load the pretrained ResNet50 model and fine-tuned for this new dataset. This library has Google inception model, Microsoft ResNet, VGG etc. model which are winners of the Imgenet challenge over the years. Anaconda package was used for this task. In jupyter notebook python language used to implement the algorithm. Tensor flow was used for creating a deep neural network. To decide on the layers to add/replace in last section of the network different combinations of the convolution, max pooling, activation and normalization layers have been used. Hyperparameters to tune were learning rate, number of kernels in convolution, kernel size, number of epoch and optimizer to be used.

Out of given 5000 images 70% of the images were used to train the model and 20% to validate the learned weights. 10% of the data then used to test the newly trained weights and to check the performance of the network. GPU cluster was used to implement this transfer learning because of the heavy computations involved for large number of epochs.

2. History

Biologist started studying mechanism of visual recognition and ability to differentiate and recognize objects in history. One of the most influential work which inspired computer vision field is by Hubel and Wiesel in 1959 [6]. Using electrophysiology, they studied mammals to see how the respond to different shapes. They observed in cat that for specific shapes shown to them, specific neuron impulses were generated. Another study by Larry Roberts in 1963 introduced to block world to visualize object in real world [7]. "The summer vision project" from MIT in 1966 tried to reconstruct visual recognition part of our system [8]. These advances in the field of machine vision blossomed many researches worldwide in these 50 years. Having said that we are still struggling to solve fundamental problem of vision. Another influential book from MIT vision scientist David Marr proposed several steps to reconstruct object in computer. In seventies, Brooks and Binford proposed method "Generalized Cylinder" [9]; Fischler and Elschalger proposed "Pictorial structure" [10]. They proposed that every object can be constructed using simple geometric structures. But all these studies were lagging in providing a concrete model which would recognize real world objects.

Scientist then started segmenting objects by observing similarities in the images. Rather than identifying what the object is segmentation differentiates different identities from background. Shi and Malik in 1997 this segmentation method [11]. Since then researches came up with face detection, pedestrian detection, object detection with different orientation algorithms. Histogram of Gradients, Deformable part model, Spatial pyramid matching are few of them [12], [13]. And then in early 2000 PASCAL Visual Object challenge was announced. Task was to detect important building blocks by object recognition. First time we had benchmark dataset to measure performance of algorithms. In 2009, Deng et. al at Stanford created a dataset called Imagenet [14]. This huge dataset consists of around 15 million images of categorized in 22 thousand different labels. Problem of overfitting due to small dataset in traditional machine learning algorithm caused the need of huge dataset. This three years of project captured most of the objects in the world. And to benchmark the progress of research in

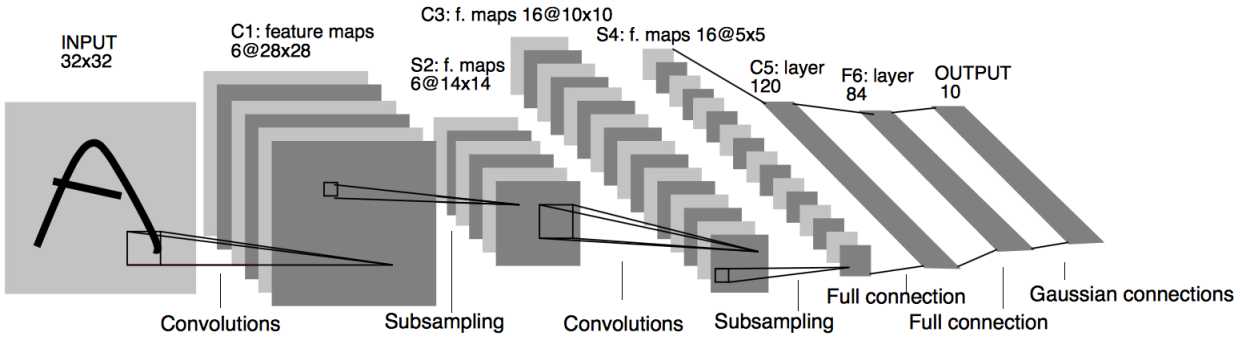


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Figure 1: Le Cun 1998

machine vision, in 2010 they rolled out and international challenge called Imagenet Large Scale Visual Recognition Challenge(LSVRC). This challenge includes 1.4 million of images with 100 classes. Since then each year error rate has been dropping significantly and neural networks are getting deeper and deeper. This challenge inspired different convolutional neural networks and deep learning techniques.

2.1. Convolutional Neural Network

In 2010, Lin et. al. won the challenge and they used NEC-UIUC network [15]. AlexNet (SuperVision) was the winner of 2010, GoogLeNet in 2014 and MSRA in 2015. Every network was hierarchical, and they used the notion of deep convolutional neural network. This idea of using CNN was inspired by work done by LeCun et. al. in 1998 [1]. But due to recent advancement in the hardware technology only now these can be implemented and tested for performance.

Concept of neural network and deep learning was first time implemented on Handwritten zip code recognition. This was developed in 1989 with three hidden layers consisting of convolution and subsampling operations followed by fully connected layer. To recognize digits final layer is classifier with 10 output units. Only first two layers were convolutional and last were fully connected layers. This article has inspired modern architecture of pooling and dropouts. Input image is 16x16 with grayscale values and output is 10-dimensional vector. Kernels used in first layer were 12 with size of 5x5 with stride 2. The configurations gave 768 hidden units in first hidden layer.

Unlike simple house pricing 2 dimensional data, an image is very high dimensional object. And image pixel values vary a lot with different light condition, orientation of camera or with zoom effect. Object recognition task should be invariant to thee changes and should detect the object without being having holistic idea of the image object. One of the simple way is to calculate mean photo and subtract it from all images to get normalized data. Similar techniques performed efficiently when used in eigen face and fischer face for face recognition. These methods are improved over the time with use of different dimensionality reduction methods and optimization technique. But still problem of finding real life objects is a challenge. Convolutional Neural Network, which is a feed-forward artificial neural network, gives surprisingly very good prediction accuracy. Typical CNN has stacked CNN layers followed by normalization and max pooling. Last layers are fully-connected and final layer is loss layer. To avoid over-fitting, caused due to multiple layers, dropout method is used.

2.2. Resnet

This is another very deep architecture developed by He et al. at Microsoft research lab for imagenet challenge [16]. This model works on the concept of residual connections. Model consists of 152 layers and achieved 3.57% top 5 error. This new design won the ILSRVC 2015 challenge also COCO 2015 challenge with good margin. Traditional idea of increasing number of layers to get better and better accuracy was not true when checked for deeper CNN. A novel technique of residual network was used wherein data from input layer is referenced directly in further layers without being unreferenced through multiple layers. This new method allows to create deeper network with simple architecture. ResNet is much deeper than VGG and yet simpler than it. There are drawbacks of having much deeper network to give better performance. Problems involved in large number of layers are discussed further in the section.

As shown in the graphs in figure 2 on the following page as number iterations increases with increase in layers, training error reduces initially but then either flattens over time or increases in some cases [16]. Also, sometimes less number of layers performs better than much deeper network. This observation holds true on training and test error. Interesting fact is that the cause of this behavior is not the overfitting but the optimization problem to train hyper-parameters. This repudiate the idea that accuracy obtained on smaller network can be used to get better performance

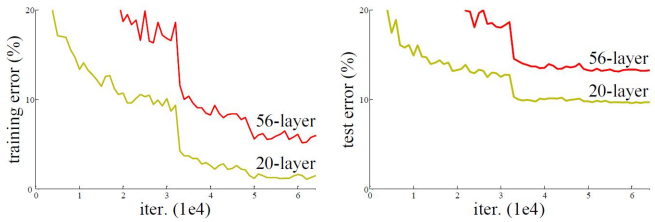


Figure 2: Deep Network Trend

by adding few networks to it. And hypothesis of better performance of deeper model should at least be as good as shallower model was rejected.

3. Methodology

In my approach I used pretrained ResNet without top layer. For small dataset, with just convolution, pooling and fully connected layer network learned the data and gave 99% accuracy on the training set but failed to perform on the testing set by giving prediction accuracy of 56%. And hence the need of data preprocessing, and regularization was subtle. Model was getting overfitted with the training data and has high variance in trained curve. Following are the layers which are then added to overcome the problem and to improve prediction accuracy on test data.

```
_________________________________________________________________
Layer (type) Output Shape Param #
=================================================
image_input (InputLayer) (None, 224, 224, 3) 0
_________________________________________________________________
resnet50 (Model) multiple 23587712
_________________________________________________________________
convNew (Conv2D) (None, 1, 1, 512) 4194816
_________________________________________________________________
banNew (BatchNormalization) (None, 1, 1, 512) 2048
_________________________________________________________________
global_average_pooling2d_1 ( (None, 512) 0
_________________________________________________________________
fc-2 (Dense) (None, 256) 131328
_________________________________________________________________
activity_regularization_1 (A (None, 256) 0
_________________________________________________________________
dropout_1 (Dropout) (None, 256) 0
_________________________________________________________________
outputLayer (Dense) (None, 7) 1799
=================================================
```

3.1. Data preprocessing and augmentation

In our data set images are different shape and size. And to create a transfer learning model on this data there are different preprocessing which can be performed for better results. These could be to bring each image in same resolution and aspect ratio, scaling, normalization, augmentation, dimensionality reduction etc. Since our dataset is very small to train this step is very crucial and important to be implemented with care. So, data preprocessing can be classified into three forms as

- Mean subtraction
- Normalization
- Dimensionality reduction

To overcome the problem of small dataset data augmentation technique is used. This generally provides more in-depth knowledge and helps in reducing manual manipulation of data to extract this information. Using extrapolation and probability techniques this can be achieved but common techniques used image analysis are as below:

- Rotation
- Mirroring
- Cropping
- Shifting channel
- Adding noise

In our analysis keras library function "ImageDataGenerator" is used to perform preprocessing and augmentation.

```
-----
train_batches = ImageDataGenerator(rescale=1./255,
rotation_range=10, width_shift_range=0.2,
height_shift_range=0.2, shear_range=0.15,
zoom_range=0.1, horizontal_flip=True,
vertical_flip=True).flow_from_directory(training_path,
target_size=(w_img,h_img), batch_size=BATCH_SIZE)
-----
```

3.2. Regularization

This technique is generally used when there is chance of high variance and overfitting. Problem of overfitting can be solved by training model on large dataset but in our case, which was not the case. This can be achieved by adding a regularization term to the optimization problem. In following equation second and term collectively acts as a regularization. This usage of L1 and L2 norm together is called as Elastic Net. While L1 norm is good for high dimensional space it fails to perform group selection when there are correlated features. On the other hand L2 norm puts constraint on values of weights but can not perform feature selection. In my approach I used combination of both called as elastic net which helped in handling problem of sparse values of weight matrix W and selecting correlated features if any.

In Keras this can be implemented as below:

```
-----
x = ActivityRegularization(l2=0.01, l1=0.01)(x)
-----
```

$$\min_{\beta\,\lambda} J(\beta) = \frac{1}{2N}\sum_{i=1}^{N}\left(f_\beta\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda_1\|\beta\|_1 + \lambda_2\|\beta\|_2$$

where,

$$\|\beta\|_p = \left(\sum_{i=1}^{N}|\beta|^p\right)^{\frac{1}{p}}$$

l_0 norm: Pseudo norm (Hard to optimize)
l_1 norm: LASSO
l_2 norm: Ridge Regression

3.3. Convolution layer

In attempt to tune ResNet model on new dataset an additional convolutional layer has been added. In keras this can

be achieved by function Conv2D. This function implements a spatial convolution on images. Parameters which needed to be tuned are number of kernels and kernel size. This function creates series of convolution kernels which then convolved with input/previous layer to produce tensor of outputs.

```
-----
x = Conv2D(512, (2, 2), name='convNew', padding =
'same')(x)
-----
```

3.4. Batch normalization

This important step invented by Ioffe Sergey, and Christian Szegedy helps in hyperparameter search problem easier [3]. This also helps in training much deeper network. For resnet, which is a deep neural network this step was added in transfer learning to improve performance.

In keras this can be implemented using "BatchNormalization". This function" Normalize the activations of the previous layer at each batch, i.e. applies a transformation that maintains the mean activation close to 0 and the activation standard deviation close to 1." [17] In our implementation it has been used as below:

```
-----
x = BatchNormalization(axis=-1, name='banNew')(x)
-----
```

3.5. Global pooling

The novel technique of global average pooling was introduced by [4]. Idea is to connect kernel convolution output to dense network via a layer which takes average of each feature map. As this helps in minimizing the overfitting it also helps in mapping convolution layer to categorical layer.

In keras this is implmeneted using "GlobalAveragePooling2D"

3.6. Dropout

In this type of regularization some of the nodes in the network are excluded in making decision of the activations. In my implementation I used 0.2 factor in dropout. In keras this can be implemented with function "Dropout". While training the neural network dropout is used but while testing on unseen data all nodes are made active to make decision.

4. Evaluation and result

In this project, model thus created has total 27 million parameters and 7 new layers are added. Steps involved in the training neural network can be briefed as shown in the algorithm 1.

Different classes used are as below:

- creamy_paste
- diced
- grated
- juiced

Algorithm 1 Pseudo Code

```
- Import required
- Initialize environment
- Data Augmentation and preprocessing
- Load pre trained model from keras
- Modify model
- Enable only new layers to learn
- Set optimizer and its parameters
- Train model for custom dataset
- Evaluate performance by checking prediction accuracy on
test data
```

- jullienne
- sliced
- whole

Every training image has been labelled with a vector of 7 values of 1 or 0. Only correct label is enabled with value of 1 and rest all are kept 0. Figure 3 shows tagging used in this project.

Figure 3: Image tagging

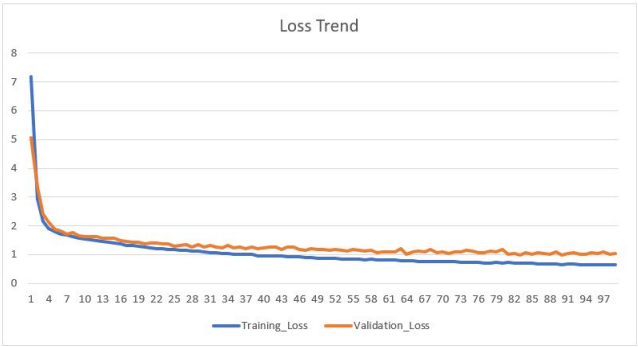
In an attempt to get good prediction accuracy on unseen data, given data is randomly split into three categories viz training, validation and Testing. Percentage split used was 7:2:1 respectively. I have tried different augmentation methods along with, varied kernel sizes, different learning rates and different kernel sizes. Table 1 on the next page shows prediction accuracy achieved on test data for these different combinations.

Table 1 on the following page shows different configurations used to improve prediction accuracy on validation data. After removing top layer from the pretrained keras model one more layer of average pooling was removed and checked for the performance of the model. First column in the table shows if this layer is used or not. Column "#new Layers" show number of new layers which are added on top of existing model.

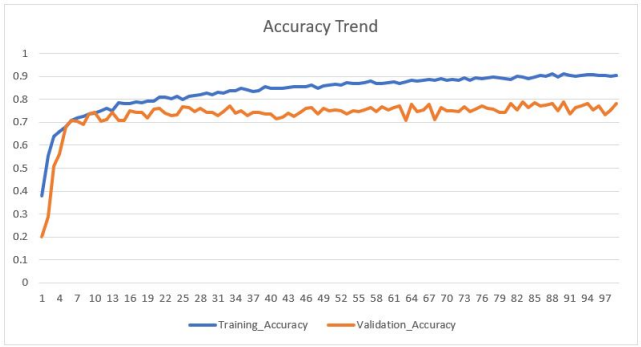
In our case we used ResNet50, pre-trained model on imagnet, in Weka. I did not use final layer; rather we

Pop_AvgPooling	Kernels	# epochs	Data Augmentation Variants	Optimizer	Learning Rate	#new Layers	Accuracy
FALSE	256	20	9	Adam	0.001	8	0.5
FALSE	256	100	9	rmsProp	0.0001	8	0.6
FALSE	256	100	4	rmsProp	0.0001	7	0.75
FALSE	256	10	4	Adam	0.0001	7	0.74
FALSE	1024	100	4	rmsProp	0.0001	5	0.65
TRUE	256	10	8	Adam	0.0001	7	0.72
TRUE	256	10	8	rmsProp	0.0001	7	0.73
TRUE	256	100	8	Adam	0.0001	7	0.76
TRUE	256	100	8	rmsProp	0.0001	7	0.77
TRUE	128	100	8	Adam	0.0001	7	0.75
TRUE	128	100	8	rmsProp	0.0001	7	0.77
FALSE	128	100	8	rmsProp	0.0001	7	0.76

TABLE 1: Analysis



(a) Loss Trend



(b) Accuracy Trend

Figure 4: Fine Tuned Model Trend

modified and added few new layers. I also checked performance of the system with one more layer layer removed after enabling "include_top" variable to FALSE. And based on prediction accuracy trend curve one model is chosen to be tested on unseen data[1]. As shown in the table "Pop_AvgPooling" is that flag which mentions if additional layer is pop or not.

4.1. Problems with training deep architecture

Due to large number of layers, there are too many parameters which need to be learned. There are chances of over-fitting the network. This problem is exacerbated if dataset is small. And for large dataset labeling them is a time and energy expensive. Moreover, increase in number of filters increases kernel computation in quadratic ratio.

- Vanishing gradient
- Exploding gradient

Degradation problem occurs when deeper network starts to converge. As it gets deeper accuracy of the model saturates and then after one point it starts degrading. It can also be observed that this problem is not because of overfitting or because of scarcity of the data.

1. Microsoft Excel is used to plot curves

5. Conclusion

It can be observed that hyperparameter optimization is very crucial in transfer learning. Even with pretrained network getting a high prediction accuracy requires fine tuning of these parameters. Also, it is necessary to have large dataset to get good feature knowledge of new dataset.

Future enhancements:

- Usage of constraints on network parameters during optimization
- Training using large number of epochs
- Using large dataset
- In depth observation of output at each layer
- Comparing performances of different optimizers

References

[1] LeCun, Yann, et al. "Backpropagation applied to handwritten zip code recognition." Neural computation 1.4 (1989): 541-551.

[2] https://github.com/deeplizard2/ Keras_Jupyter_Notebooks/blob/master/CNN.ipynb

[3] Ioffe Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." International conference on machine learning. 2015.

[4] Lin, Min, Qiang Chen, and Shuicheng Yan. "Network in network." arXiv preprint arXiv:1312.4400 (2013).

[5] https://www.youtube.com/watch?v=QaCSzIzd3Vs&list= PLd9i_xMMzZF7eIjnVuPxggYuGPg5CnA1l

[6] Hubel, David H., and Torsten N. Wiesel. "Receptive fields of single neurones in the cat's striate cortex." The Journal of physiology 148.3 (1959): 574-591.

[7] Roberts, Lawrence Gilman. "Machine Perception of Three-dimensional Solids." Diss. Massachusetts Institute of Technology, 1963.

[8] Marr, David. "Vision." The MIT Press, 1982.

[9] Brooks, Rodney A., and Creiner, Russell and Binford, Thomas O. "The ACRONYM model-based vision system. " In Proceedings of the 6th International Joint Conference on Artificial Intelligence (1979): 105-113.

[10] Fischler, Martin A., and Robert A. Elschlager. "The representation and matching of pictorial structures." IEEE Transactions on Computers 22.1 (1973): 67-92.

[11] Shi Jianbo, and Jitendra Malik. "Normalized cuts and image segmentation." Pattern Analysis and Machine Intelligence, IEEE Transactions on 22.8 (2000): 888-905.

[12] Viola Paul, and Michael Jones. "Rapid object detection using a boosted cascade of simple features." Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on. Vol. 1. IEEE, 2001

[13] Dalal Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. Vol. 1. IEEE, 2005.

[14] Deng Jia, et al. "Imagenet: A large-scale hierarchical image database." Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. IEEE, 2009

[15] Lin Yuanqing, et al. "Large-scale image classification: fast feature extraction and SVM training." Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. IEEE, 2011

[16] He Kaiming, et al. "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition." arXiv preprint arXiv:1406.4729 (2014).

[17] https://keras.io/