UNIVERSITY NAME (IN BLOCK CAPITALS)

Weather Classification

by

Junjie Zhang

A thesis submitted in partial fulfillment for the
degree of Master

in the

Chunhua Shen
School of Computer Science

November 2015

Declaration of Authorship

I, Junjie Zhang, declare that this thesis titled, 'Weather Classification' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

Abstract

Scene classification is an important field in computer vision. For similar weather condition, there are some obstacles for extracting features from outdoor images. In this paper, we present a novel approach to classify cloudy and sunny weather. Inspired by recent study of deep multi-layer neural network and spatial pyramid polling, generate a model based on imagenet dataset. Starting with parameters trained from more than 1 million images, we fine-tune the parameters. Experiment demonstrates that our classifier can achieve the state of the art accuracy.

Acknowledgements

The acknowledgements and the people to thank go here, don't forget to include your project advisor...

Contents

List of Figures

List of Tables

Abbreviations

LAH **L**ist **A**bbreviations **H**ere

Physical Constants

Speed of Light $c = 2.997\ 924\ 58 \times 10^8\ \mathrm{ms}^{-\mathrm{S}}$ (exact)

Symbols

a	distance	m
P	power	W (Js^{-1})
ω	angular frequency	$rads^{-1}$

For/Dedicated to/To my...

Chapter 1

Introduction

1.1 Overview

Computer is one of the most significant inventions in history. It provides huge power in data processing field, like scientific computation. Also, computer system can aid human being in daily life, for example driverless vehicles. However, human brain still has compelling advantage in some fields, like identifying our keys in our pocket by feel. The complex processes of taking in raw data and taking action based on the pattern are regarded as pattern recognition. Pattern recognition has been important for people daily life for a long period and human brain has developed an advanced neural and cognitive system for such tasks.

Weather classification is one of the most important pattern recognition tasks which relates our work and lives. There are several major kinds of weather conditions, like sunny, cloudy, rainy. And people can classify them easily through eyes. However, it is not a easy job for machines, especially in computer vision literacy.

In this thesis, I describe an approach to this problem of weather classification based upon the new big trend in machine learning, and more precisely, deep learning. It differs with feature detectors method that extracts features manually and training a model to do classification.

Compared to shadow learning which includes decision trees, SVM and naive bayes, deep learning passes input data through several non-linearities functions to generate features and does classification based on the features. It generates a mapping and finds a optimist solution.

1.2 Statistical Pattern Recognization

In the statistical approach, the pattern is represented in terms of d dimensional feature vector and each element of the vector describes some subjects of the example. In brief, three components are essential to do statistical pattern recognization. First is a representation of the model. Second is an objective function used to evaluating the accuracy of the model. Third is an optimization method for learning a model with minimum errors.

1.3 Artificial Neural Networks

Artificial neural networks(ANNs) were proposed in the mid-20th century. The term is inspired by the structure of neural networks in the brain. It is one of the most successful statistical learning models used to approximate functions. Learning with ANNs yields an effective learning paradigm and has achieved cutting-edge performance in computer vision.

The single-layer perceptron has shown great success for a simple model. For increasing complex models and applications, multi-layer perceptron has exhibited the power of features learning. With hardware developing At the same time, the demanding of more efficient optimizing and model evoluation methods is needed for BIG DATA.

Recent development in ANNs approaches have giantly advanced the performance of state-of-the-art visual recognition systems. With implementing deep convolutional neural networks, it achieves top accuracy in ImageNet Challenge. The model has been used in related fields and succeeds in competition.

1.4 Weather Classification

Weather classification is an important job in daily life. In this thesis, we are focused on two class weather conditions, sunny and cloudy.

There are some obstacles in front of weather classification. First, because inputting pixels are independent and high dimension, say a 500x500 RPG image means 750000 pixels, computation is huge. Second, some simple middle level image characters are difficult to machines, like light, shading, sun shine. It is still not easy to detect these characters with high accuracy. Thirdly, there are no decisive features, for example brightness and

lightness, to classify scene. For example, sun shine can be both found in sunny and cloudy weather. Last but not least, outdoor images are various background.

Chapter 2

Introduction

2.1 Single-Layer Networks

Artificial neurons were introduced as information processing devices more than fifty years ago[1]. Following the early work, perceptrons were deployed in layers to do pattern recognition job. A lot of resource was invested in researching capability of learning perceptrons theoretically and experimentally. As shown in Figure 2.1, a perceptron computes a weighted summation of its n inputs and then thresholds the result to give a binary output y. We can treat n input as an vector with n elements, and represent the vector as either class A(for output +1) or class B(for output -1).

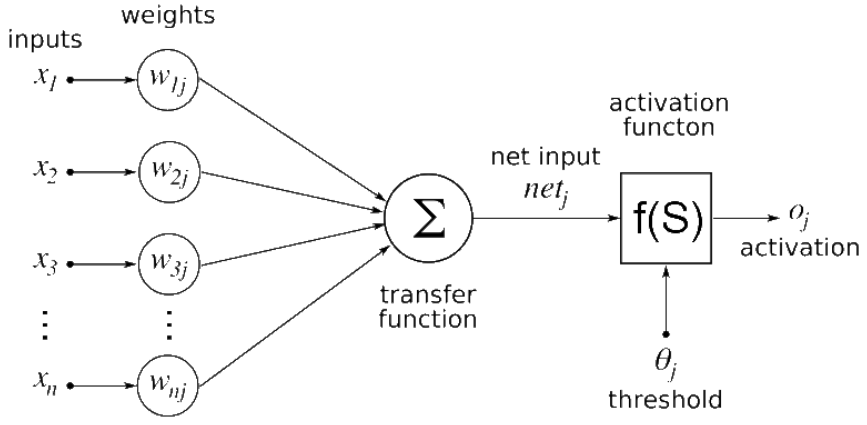


FIGURE 2.1: Diagram of a perceptron.

Each output is updated according to the equation:

$$y_i = f(h_i) = f\left(\sum_j w_{ij} x_j\right) \tag{2.1}$$

where y_i is the ith output, h_i is the net input into node i, The weight w_{ij} connects the ith output and the jth input, and x_j is the jth input. The function $f(h)$ is the activation function and usually makes up the form

$$f(h) = sign(h) = \begin{cases} -1 & h < 0 \\ 1 & h \geq 0 \end{cases} \tag{2.2}$$

We can also represent 2.2 in vector notation, as in

$$y = f(h) = f(\mathbf{w} \cdot \mathbf{x}) \tag{2.3}$$

where $\mathbf{w}$ and $\mathbf{x}$ can be regarded as $nx1$ dimensional column vectors, and n is the number of input data. The term $w \cdot x$ in 2.3 constructs a $(n-1)$-dimension hyperplane which passes the origin. The hyperplane can be shifted by adding an parameter to 2.1, for example

$$y = f(h) = f(\mathbf{w} \cdot \mathbf{x} + \mathbf{b}) \tag{2.4}$$

We can have the same effect by putting a constant value 1 and increase the size of x and w by one. The extra weight w with fixed value 1 is called *bias weight*; it is adaptive like other weights and provides flexibility to hyperplane. Then we get:

$$y = f(h) = f(\sum_{i=0}^{n} w_i x_i) \tag{2.5}$$

The aim of learning is to find a set of weights w_i so that:

$$y = f(\sum_{i=0}^{n} w_i x_i) = 1 \quad x \in ClassA$$
$$y = f(\sum_{i=0}^{n} w_i x_i) = 0 \quad x \in ClassB$$

2.2 Multi-Layer Networks

Single-layer networks have some important limitation in terms of the representing range of functions. We are seeking to learn the nonlinearity as the linear discriminant.To improve the representation capability, we can stack layers networks. This is the approach of multilayer neural networks. Multilayer neural networks implement linear discriminants via mapping input data to a nonlinear space. They can use fairly simple algorithms to learn form of the nonlinearity from training data.

In the thesis, we limit multi-layer networks in subset of feedforward networks. As feedforward networks can provide a general mechanism for representing nonlinear functional mapping between a set of input data and a set of labels. The 2.2 is a feedforward network having two layers of adaptive weights.

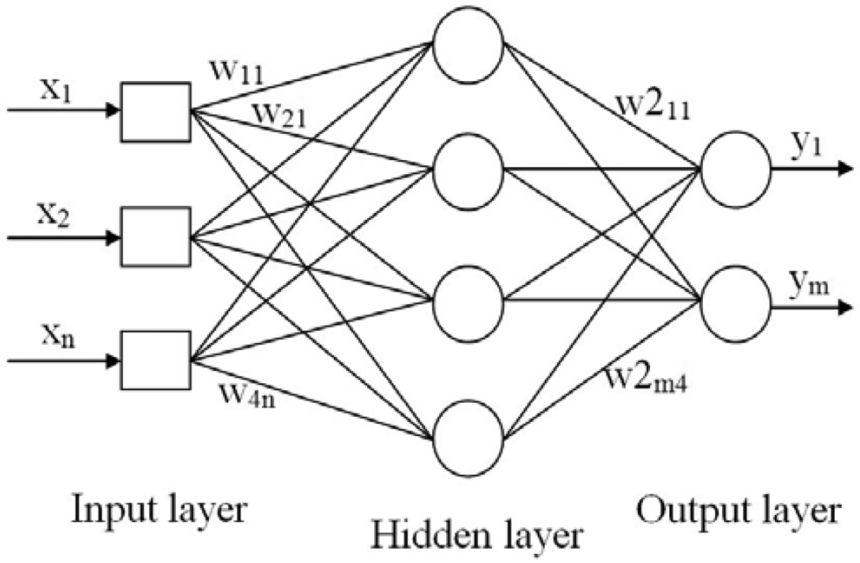


FIGURE 2.2: Diagram of a feedford neural networks.

In the example, the middle column perceptrons act as hidden units. The network has n inputs, 4 hidden units and m output units. The network diagram represents the function in the form

$$y_m = \hat{f}\Big(\sum_{j=0}^{m} w_{j4}^{(2)} f\big(\sum_{i=0}^{n} w_{4i}^{(1)} x_i\big)\Big) \tag{2.6}$$

In the 4.2, outer activation function could be different with the inner one.

There are some choices for activation functions, sigmoid and tanh are related and can provide high capability with continuous input data. Logistic activation function sigmoid can be represented as

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.7}$$

Its outputs lies in range $(0, 1)$. We can do a linear transformation $hatx = x/2$ on input data and a linear transformation $haty = 2y - 1$ on the output. Then we can get an equivalent activation function tanh which can be represented as

$$f(x) = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.8}$$

The three layer neural network is capable of approximating any function with enough hidden units which means the networks with two layers of weights and sigmoid nonlinearities can provide any accuracy in classification problems.

2.3 Backpropagation

Multi-layer neural networks can represent mapping from input data to output classes. How to learn a suitable mapping from training dataset? And there is no explicit mapping between output and hidden units. This will be resolved by a popular learning algorithm-backpropagation.

Because networks have differentiable activation functions, the activation of output units can be propagated to hidden units with regard to weights and bias. If we have an error function, we can evaluate derivatives of the error and update the weights to minimize the error function via some optimization methods.

Backpropagation can be applied to find the derivatives of an error function related to the weights and biases in the network via two stages. First, the derivatives of the error functions, for instance sum-of-squares and Jacobian, with respect to the weights must be evaluated. Second, a variety of optimization schemes can be implemented to compute the adjustment of weights based on derivatives. After putting data forward propagation through the network, we can get the output result. It updates weight changes based on grandient descent. Suppose the network has i inputs, h hidden units and k outputs. The update equation can be represented as

$$w(j+1) = w(j) + \Delta w(j) \tag{2.9}$$

where $\Delta w(j)$ defined as

$$\Delta w(j) = -\eta \frac{\partial E}{\partial w} \tag{2.10}$$

For the hidden to output layer weights

$$\Delta w(jk) = -\eta \frac{\partial E}{\partial w_{jk}} = -\eta \delta_k y_j \tag{2.11}$$

where

$$\delta_k = \frac{\partial E}{\partial a_k} = (y_k - t_k) y_k (1 - y_k)$$

For the hidden layer weights

$$\Delta w(ij) = -\eta \frac{\partial E}{\partial w_{ij}} = -\eta \delta_j y_i \tag{2.12}$$

where

$$\delta_j = \frac{\partial E}{\partial a_j} = \sum_k \delta_k w_{jk} y_j (1 - y_j)$$

The δ_j of a hidden unit is based on the δ_ks of output units to which it links. To minimise the error function$E$ with gradient descent, it needs the propagation of errors backwards.

2.4 Training protocols

In supervised learning, we have training dataset which is data with labels. We can use the neural networks to find the output of the training data and adjust the weights to optimal values. There are mainly three types of training protocols, stochastic, batch and online training. In stochastic training, we randomly choose samples from training dataset and update weights every time depending on output of neural networks. In batch training, we use some samples and pass them through network, then update weights. In online training method, each sample of training dataset is used once and weights are updated each time. We usually call one time of passing all training dataset through neural networks one epoch.

2.5 Stochastic Gradient Descent

Because weights space in neural networks is continuous, training can reach the optimal weights value through optimization algorithms, which means minimizing loss value of function

$$L(f_w) = \sum L(y, f_w(x)) \tag{2.13}$$

Gradient descent is a method which starts from a random point, then moves to a nearby point that is downhill repeatly. It converges on a minimum possible loss.

Stochastic gradient descent is a subtype of gradient descent. It only considers a single training point and move to nearby point based on

$$w = w - \eta \Delta L(w) = w - \eta \sum_{i=1}^{n} \Delta L_i(w) \tag{2.14}$$

where η is the learning rate and $L_i(w)$ is the value of the loss function at the i^{th} sample. Although stochastic gradient descent does not guarantee convergence, it is fast.

2.6 Convolutional Neural Networks

Convolutional Neural Networks[2] are widely applied in image data and they achieved top rank in image classification competetion[3]. Convolutional neural networks have

three types of layers, convolutional layer, pooling layer and fully connected layer. At the output of the network, a softmax layer is used to do classification job. Convolutional layer is used to detect invariants in locat receptive fields. Pooling layer is downsampling feature maps. Fully connected layer makes the output of each unit an activation of dot product of all inputs.

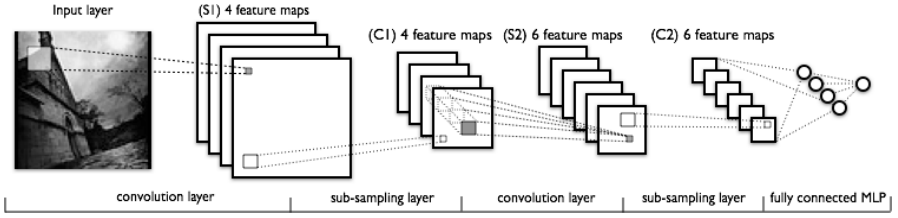


FIGURE 2.3: Diagram of LeNet.

2.7 Spatial Pyramid Match

Spatial pyramid match[4] is used to classify high-level semantic attributes, based on low-level features. The method subdivides a image in several different levels of resolution and counts features falling in each spatial bin. It extends bags of features and derives spatial information from images.

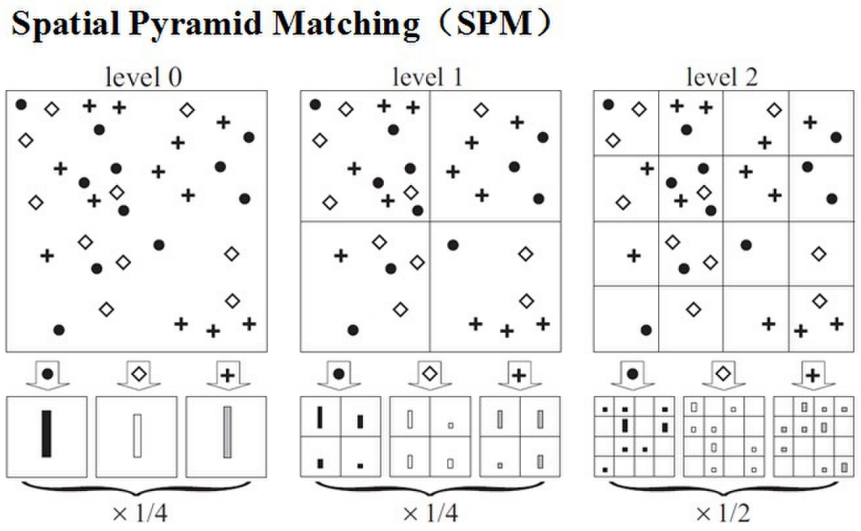


FIGURE 2.4: Diagram of Spatial Pyramid Match.

2.8 Transfer Learning

In machine learning literature, transfer learning[5] focuses on storing knowledge from one domain and applying it to a related problem. It has two benefits. One is saving

time to build a model from scratch up. Another is saving effort to collect training data.

A domain with two components, a feature space and a marginal probability distribution, can be represented

$$D = \{\chi, P(X)\} \tag{2.15}$$

where χ is feature space and $P(X)$ is the marginal probability distribution.

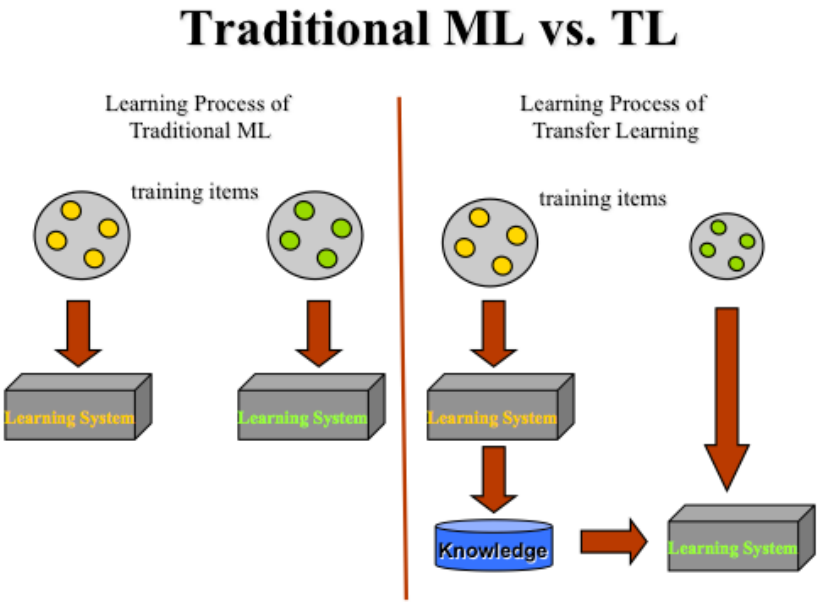


FIGURE 2.5: Diagram of Transfer Learning.

Chapter 3

Methodology

3.1 Dataset

There are over 15 million labeled images in ImageNet database belonging to about 22,000 categories. The images were collected from the Internet and labeled manually. From 2010, an annual competition named the ImageNet Large-Scale Visual REcognition Challenge (ILSVRC) has been held. The competition uses a subset of dataset which is 1000 images in 1000 categories. Totally, there are over 1 million training images and 50,000 validation images and 150,000 images for testing.

FIGURE 3.1: 2 Figures from ImageNet

For weather dataset, it contains 10,000 images for two categories evenly, cloudy and sunny. They are from three sources, Sun Dataset[6], Labelme Dataset[7] and Flickr. They were classified manually and similar images are removed. There are no unambiguous images.

FIGURE 3.2: 2 Figures from Weather Dataset

The two datasets are different. ImageNet dataset is for object classification and weather dataset is for scenes classification. Because the two datasets consists of different resolution images, we have to resize them to a fixed resolution of $256x256$ for constant dimension input for neural networks. To train the model with ImageNet images, we resize shorter side of images to length 256 and then cropped out a $256x256$ image from center.

3.2 Data Argument

The model has about 60 million neurons, and it is essential to avoid overfitting. One of methods is to do data argument. Dataset is artificially enlarged via image transformations and horizontal reflections. For each $256x256$ image, the network extracts five $224x224$ patches from four corners and center and reflects them horizontally. Hence, there are 10 patches for 1 image in total.

3.3 Spatial Pyramid Pooling

To improve the capability of generalization, a spatial pyramid pooling layer is deployed behind the fifth convolutional layer. A set of bin sizes are set to discern different size local information from output of the fifth convolutional layer. A sliding window pooling is implemented on the feature maps after the fifth convolutional layer. Assuming dimension of feature map is axa and bin size is n, each window size is $\lceil a/n \rceil$ and stride size is $\lfloor a/n \rfloor$ where $\lceil\rceil$ and $\lfloor\rfloor$ denote ceiling and flooring operations. For l level pyramid, there are l spatial pyramid pooling layers, The the l layers will be concatenated into a fully connected layer. The bin sizes are 1, 2, 3 and 6.

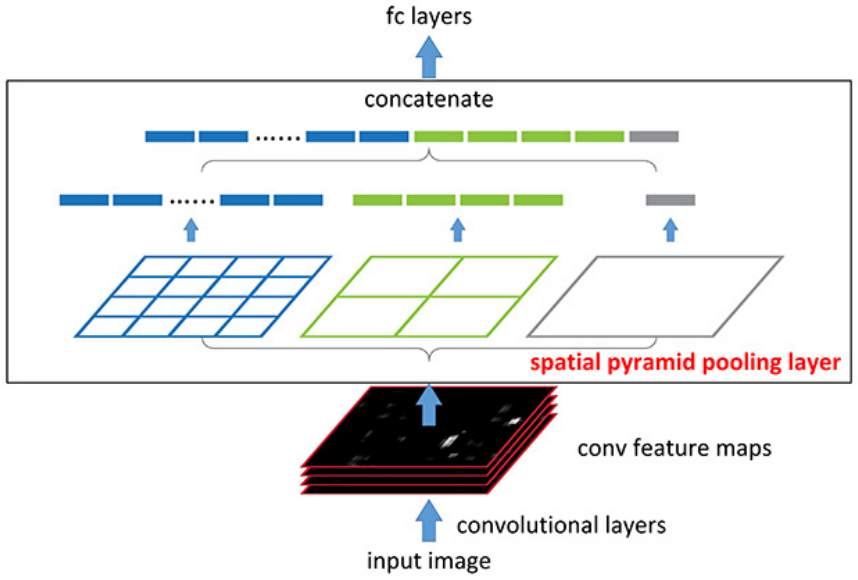


FIGURE 3.3: Diagram of Spatial Pyramid Pooling layer

With implementing spatial pyramid pooling layers in convolutional neural networks, all contribution of different scales are considered.

3.4 Convolutional Neural Networks Architecture

Due to the significant performance of AlexNet [3] deep convolutional neural networks, we train a model based on it. The architecture of the network has seven hidden adaptive layers-five convolutional and two fully connected layers. The network is very deep and

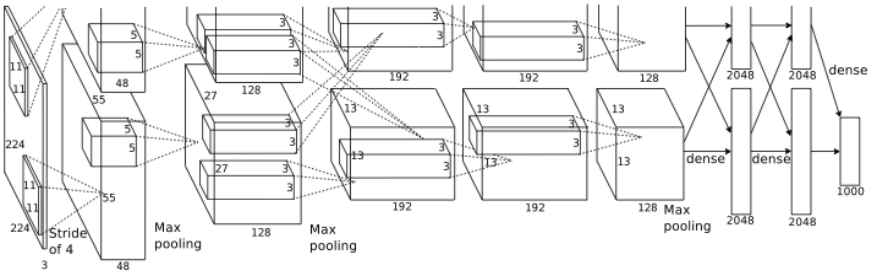


FIGURE 3.4: Architecture of AlexNet

the early layers are split over on two GPUs. It is able to give 62.5% accuracy rates with one prediction and 83% accuracy rates with five predictions.

The network replaces each neuron's outputs nonlinearity function f from $f(x) = tanh(x)$ or $f(x) = (1 + e^{-x})^{(} - 1)$ to Rectified Linear Units(RELUs)[8] which can accelerate learning speed several times faster than $tanh$ function.

In the network, there are three types of layers and they play different roles in the model. The input data dimension is $224x224x3$ which means the raw pixel value stored in a width 224, height 224 and with three color channels R,G,B matrix. In the first convolutional layer, there are 98 kernel of size $11x11x3$ with stride size 4 and outputs are 96 neurons. A max pooling layer downsamples the spatial dimensions. The second convolutional layer filters output of previous pooling layer with kernel size $5x5x48$. The third convolutiona layer owns 384 kernels of $3x3x256$ and receives outputs of the second convolutional layer. The fourth convolutional layer has 384 kernels of size $3x3x192$ and the last convolutional layer has 256 kernels of size $3x3x192$. Each fully connected layers have 4096 units. At the end of network, there is a softmax layer with 1000 outputs.

Layer Name	Layer Description
Input	224 x 224 RGB image
$Conv_1$	11 11 conv 96 ReLU units, stride 4
$Pool_1$	3 x 3 max pooling, stride 2
$Conv_2$	5 5 conv, 256 ReLU units, stride 1
$Pool_2$	3 x 3 max pooling, stride 2
$Conv_3$	3 3 conv, 384 ReLU units, stride 1
$Conv_4$	3 3 conv, 384 ReLU units, stride 1
$Conv_5$	3 3 conv, 256 ReLU units, stride 1
$Pool_5$	3 x 3 max pooling, stride 2
SPP	bin size 1,2,3,6
fc_6	fully connect, ReLU 4096 units
fc_7	fully connect, ReLU 4096 units
fc_8	fully connect, ReLU 1000 units
$softmax$	1000 way softmax

TABLE 3.1: Parameters of model

Chapter 4

Experiment

We set up experiment environment on a desktop which has hardware i7 CPU, 8G RAM and a GeForce GTX 770 and software operation system Ubuntu Linux 14.04. We train models in Caffe[9] which is a open source framework and develop spatial pyramid pooling layer in the architecture.

4.1 Training Neural Network

We trained model with the 1000-category ImageNet2012 on deep learning framework CaffeNet [9] on a We use the base network architecture of fast(smaller) model [10] which achieved an excellent result in 2013 ImageNet Competition. We implement SPP with CUDA C language and deploy them before the first fully-connected layer.

In experiment, a subset of ImageNet dataset, which contains 1.2 million labelled high-resolution images depicting 1000 object categories for training and 50,000 validation images, is used as training dataset. ImageNet contains of multi-scale and some grey images which are converted to a constant input dimensionality. Then images are resized to $256x256$. And for grey images, they are combined it triple to minic a RGB image. Model is trained on the raw RGB values of pixels. Activation function is chosen Rectified Linear Units(ReLUs), which allows for faster and effective training of deep neural networks.

$$f(x) = max(0, x) \tag{4.1}$$

The network model **??** is trained with stochastic gradient descent as backpropagation learning rule, with a batch size of 128, and momentum of 0.9. The weights are initialized

from a 0 mean Gaussian distribution with standard deviation 0.01 in each layer. The neuron biases, in the $Conv_2$, $Conv_4$, $Conv_5$ and fully connected layers, are initialized with the constant 1, while biases in other layers are initialized with constant 0.

The learning rates are set equally for all layers. It was initialized at 0.01 and decreases with stepdown policy which means that it would drop by a factor of 10 after 100000 iteration. In total, the learning rate drops 3 times and accuracy stops after 370K iterations.

The training was regularised by dropout and weight decay. Dropout regularisation is implemented in the first two fully-connected layers and dropout ratio is set to 0.5. The neurons which are dropped out output zero and do not participate in backpropagation. Therefore, the neural network samples a different architecture each time. It greatly decreases complex co-adaptations of neurons because a neuron cannot depend on the existence of distinct other neurons. For weight decay ϵ, it is set to 0.0005 which means , after each weight update, the new weight is shrunk according to

$$w^{new} = w^{old}(1 - \epsilon) \tag{4.2}$$

4.2 Fine-tuning Model

As mentioned previously, the CNN network is pre-trained with aim of classifying objects in images. However, the target task is to classify weather scene and the pre-trained model contains more than 60 million parameters which are too high for training target model from scratch up, because the target dataset has only 10000 images in total. With the excellent accuracy achieved from previous works about transfer learning, it is a good approach to fine-tune previous model. The previous model does a job to classify 1000 categories and the target model does for two class classification.

Fine-tuning transfers weights of each layers in previous model to new one except the last fully connected layer. The last layer is taken over by a new one which contains the same amount of neurons as class number in the dataset, say two, and is initialized with random weights. One advantage of fine-tune is that it highly reduces the probability of overfitting while training with small dataset. The other advantage is the existing weights are close to optimal values and the gradient descent algorithm can converge quickly.

In the problem, we want to classify sunny and cloudy images, then the last layer of the previous architecture, $fc8$, is taken place by a layer with two neurons, one for sunny and one for cloudy. The model, trained using ILSVRC2012 dataset, is used to initialize most weights in the neural network for the fine-tuning experiment. At the same time,

1000 images are used to evaluate model. The model is fine-tuned in 10000 iterations which means that total training images are fed into CNN $\frac{128x10000}{9000} = 142$ times. The initial base learning rate is 0.001 and the rate is divided by 10 every 10 epochs. Because the weights in $fc8$ are randomly initialized which means they are not close to final optimization value, the learning rate of $fc8$ is 10 times of the base learning rate in order to converge faster.

To minimise risk of overfitting, data augmentation is introduced in to experiment. Different version of patches are generated from each image via simple transformations, for example flipping and cropping. Some practical methods have been implemented and increase accuracy [3]. We do this job by increasing the spatial generalization capability of the model by cropping 5 different patches from four corners and center of images, and flipping them. Totally, we generate 10 different images with the same label.

4.3 Companion Experimental

To compare the performance of fine-tuned model, we do some a experiment with other method. We use a pre-trained model with AlexNet architecture and extract features from fc7. Then we apply SVM on the features and learn a classifier to do weather classification.

4.4 Experimental Result

The results in table4.1 shows that supervised pre-training model have excellent generalization capability.

Methods	CNN+SVM	SPP+SVM	Finetune on CNN	Finetune on SPP
Accuracy	84.8%	82.1%	93.1%	93.98%

TABLE 4.1: CNN means with original trained model, SPP means model deployed with spatial pyramid pooling layer

The fine-tuning process is very quick, and accuracy rate convergences to over 90%. After 12 epoch, the accuracy rate exceeds 90%.

No single image dataset can totally capture variation in natural images, so even ImageNet is biased in some fields. Then pre-training may be make the CNN to overfit and then damage model generalization capability. To learning the process of ImageNet pre-training, we look into the effect of pre-training period on generalization performance

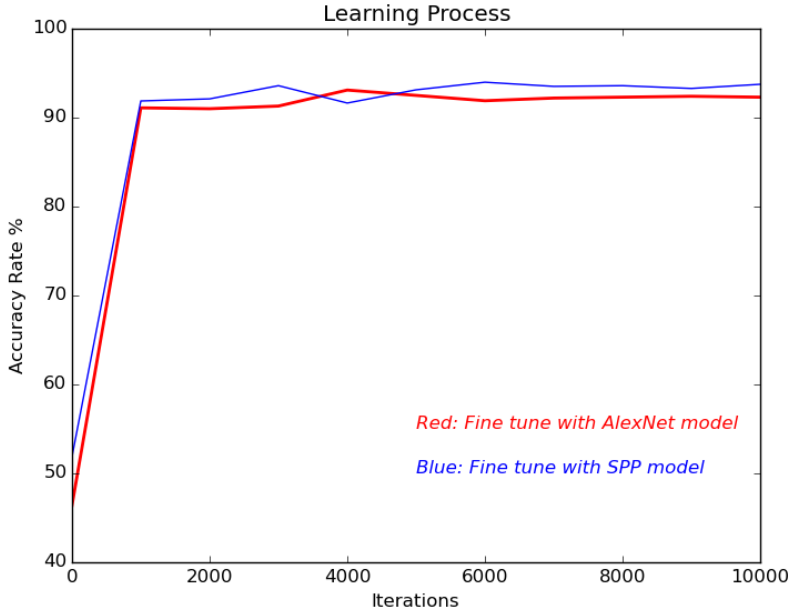


FIGURE 4.1: Finetune Process

with and without fine-tuning. In figure 4.6, we can find that longer pre-training improves performance.

To have a more detailed information of fine-tuning process, we can have a closer look at training loss. We plot the curve of first 200 iterations and loss value.

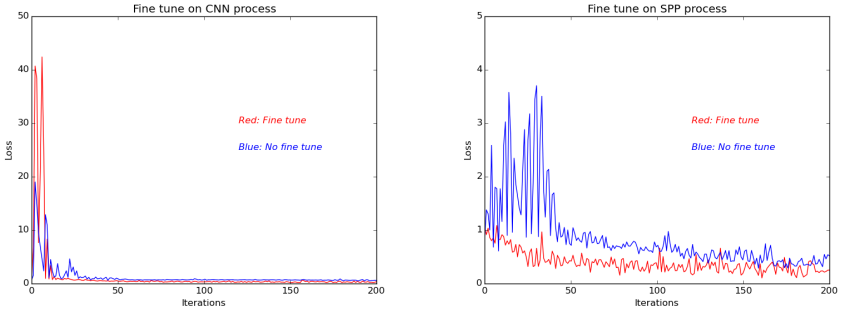


FIGURE 4.2: Finetune Process

From figure 4.2, we can find that the fine-tuning procedure produces a more smooth loss function curve and ceases with a better loss. In the left figure, the loss value of fine tune procedure is higher than no fine tune process at the beginning, then it shrinks sharply and keeps lower. In the other figure, we can find that starting loss value are both less than in the left figure. And the loss values of fine tune are less than no fine tune process. We can conclude that generally fine tune is a effective approach to reduce loss value and fine tune on SPP model can achieve a better result than CNN model.

4.5 Architecture Analysis

Although CNNs have excellent performance in computer vision field, there is still a long way to understand insight working mechanism. In order to have more insight about the reason why deep layers can increase performance of visual sentiment classification, we will do some analysis on fine-tuned network.

The outputs of each layers have been treated as visual descriptors [**?**], while the vectors are composed by outputs of neurons in previous layer. While the lower layers encode low-level features, top layers have been used to capture high-level information. In other words, we can train classifiers basing on features extracted from each layer.

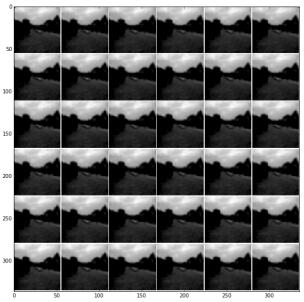
In fully-connected layer, units can be represented as d dimensional vectors, and no further manipulation is needed. The layer takes all outputs of previous layer ,for example CONV, POOL, NORM,whose feature maps are multidimensional, and flats the feature maps into d dimensional vectors. The outputs of fully connected layer are feed into a classifier. Two types of linear classifiers are used. They are linear Support Vector Machine and Softmax.

We can have a look for the feature maps after each convolutional layers.

width=0.4

(a) b

FIGURE 4.3



width=0.4

(a) b

FIGURE 4.4

FIGURE 4.5: A cloudy image

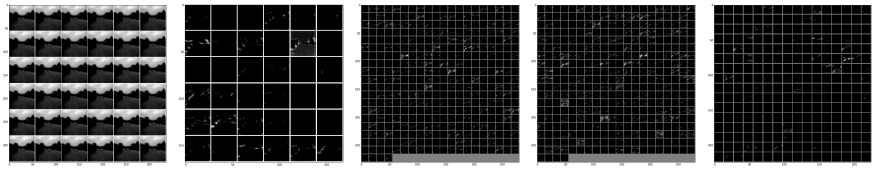


FIGURE 4.6: This is a raw cloudy image and

Appendix A

Appendix Title Here

Write your Appendix content here.

Bibliography

[1] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[2] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[4] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[5] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359, 2010.

[6] Bryan C Russell, Antonio Torralba, Kevin P Murphy, and William T Freeman. Labelme: a database and web-based tool for image annotation. *International journal of computer vision*, 77(1-3):157–173, 2008.

[7] Jianxiong Xiao, James Hays, Krista Ehinger, Aude Oliva, Antonio Torralba, et al. Sun database: Large-scale scene recognition from abbey to zoo. In *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*, pages 3485–3492. IEEE, 2010.

[8] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

[9] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[10] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013. URL http://arxiv.org/abs/1311.2901.