Plant Leaf Types classification with Deep Neural Networks

Prasanna Venkatesh Parthasarathy

1

{ppartha@masonlive.gmu.edu}

1. Introduction

Recognizing the types of leaves automatically from their images has application in species discovery, plant taxonomy, weeds identification etc. The main problem in this image classification task lies in the subtle differences between leaf species images. Also, the number of images available for training is relatively less compared to the standard experimental datasets like CIFAR due to the large variety of available plant species, and often, images from various sources are considered for the problem.

In this work, I describe about the exploration with this problem by using a standard ConvNet architecture and ConvNet architecture based on weights initialized by PCA approach.

2. Background

Research on identifying plant leaf types is a decade old problem where initial approaches involved choosing hand-crafted features. Popular image feature extraction techniques from Computer Vision domain like HOG, SIFT have also been applied for this problem. Jassmann et al [2], used ZCA whitening approach in the pre-processing step to make the features in the image less redundant. Most of these techniques follow a process of applying standard SVM or Neural Network classifier design.

Huge success of deep learning in computer vision challenges like ILSVRC[4] [7] have enabled researches to apply popular ConvNet architectures to this problem. Learning CNNs from few samples is realized in the litreature as either a sort of tranfer learning or as a layer-wise unsupervised pre-trainng. By transferring knowledge from different tasks, neural networks have shown to be able to learn classes faster and from less training data[9]. Instead of transferring knowledge from a domain specific task, generic pattern matching tasks generated from unlabeled data have also been used to improve generalization performance on small datasets.

Unsupervised pre-training has shown astonishing results on various network architectures and datasets. It places the network in a beneficial state for a supervised refinement and guides the learning process towards a minimum, that provides better generalization capabilities compared to minimar achieved from random starting points[1].

In contrast to hand-crafted feature/classifer combinations which are widely used for vision tasks, CNNs form an end-to-end trainable hybrid feature-extraction/classification architecture. In contrast to fully connected nerual networks, the extensive use of shared weights reduces the number of degrees of freedom without the loss of expressive power, which makes CNNs easy to train with plain gradient descent. CNNs have veen shown to be very competitive for visual recognition tasks [4][7], even outperforming human performance on some datasets.

3. Dataset

Swedish Leaf Dataset[8], which is collection of images from 15 tree classes. These are clean images, which is characterized with well aligned leaf on contrasting background, with little or no variations of luminance or color.

Name	Species/Classes	Samples
Swedish Leaf Dataset	15	1125 samples. 75 samples per class.

Figure 1. Examples of species with notable differences

4. Approach

I follow the general approach followed for any image classification task with variations in feature selection and model. The less noisy nature of the data does not require any intensive pre-processing techniques to be done. All the images are taken in a cleaner and white background with no override of any color intensities.

The core idea in this work is application and systematic evaluation of different classification methods to produce better results and try to reason them. A simple linear classifier, a dense neural network, CNN and CNN with PCA initalized weights [10] are the methods that are considered in this work.

Principal Component analysis (PCA) can reduce the redundancy of input data, eliminate possible correlation and extract the most relevant feature vectors for data changing directions. To ease the gradient diffusion problem in CNN, PCA is used to the unsupervised extraction of the input image eigenvectors, then selecting the main eigenvectors to initialize the weights of convolution kernels. Such kind of initialization can reduce the effect of gradient diffusion problem by bad initialization parameters. It can also enhance the classification results.

5. Experimental Design

5.1. Pre-Processing

The input dataset contains images of different sizes and shapes. The input dimensions to image classification machine learning problem are $width * height * channels$ where channels represent the values of RGB color intensities. A typical high resolution image can easily scale of thousands of dimension which will make the classification process complex and computationally ineffective. To reduce the dimensions and preserve the core chunk of the actual data, all the images are cropped and re-sized to a standard shape of $50 * 50 * 3$.

Data is also normalized such that the values of intensity ranges from $[0, 1]$ which preserves actual distribution and avoids results of larger values due to multiple calculation

of vector products. To perform well, deep learning algorithms needs lot of images to train on. Data augmentation is an automatic way to boost the number of different images that will be used to train Deep learning algorithms. For each image in the input data different variations are produced by performing rotation, shearing and image shifting.

For every experiment, train-test split ratio of $75:25$ is considered such that classifier is evaluated based on the testing performance on completely unseen data.

5.2. CNN

I started by training ConvNet classifier from scratch, closely following VGGNet [6] architecture and guidelines below:

- Convolutional layer learns features from general to specific, giving more layers helps with the transition.
- Dropout reduces overfitting
- Well designed pooling to reduce dimensionality

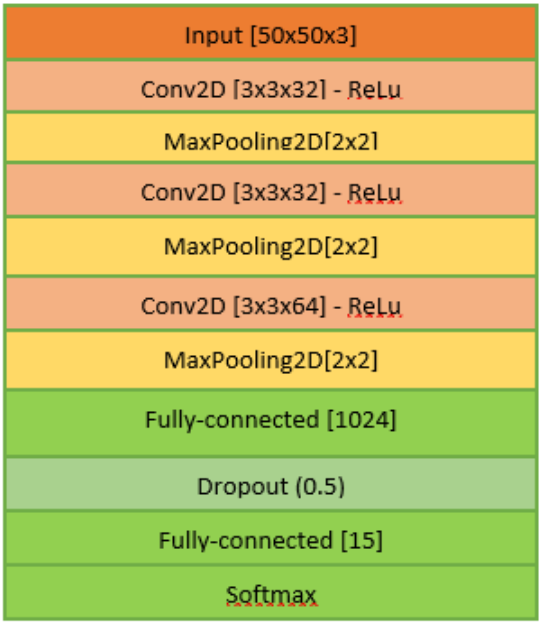

Figure 2. ConvNet Architecture

A deeper ConvNet architecture with more aggressive filters is needed to extract the features more efficiently and to deal with noise and variation of datasets. It appeared from the experiments that for the current dataset in hand, three or even two layers of convolution were to produce good results. In this customized CNN, the Convolution layers are initialized with weight matrix whose values are obtained from a random uniform distribution.

Each Convolution follows a ReLu $[f(x) = max(0, x)]$ activation and neurons in fully-connected layer are activated with softmax. Usage of ReLu induces non-linearity and combinations of ReLu are also non-linear which essential for learning a non-linear boundary using backpropagation algorithm. ReLu is also computationally less expensive that functions like sigmoid or tanh because of the simpler math involved in the calculation.

Research has shown that ReLus result in much faster training for larger networks and is efficient [5].

To produce the required output of classification problem, activations like sigmoid is required at the fully-connected layer of the network. The multi-class version of sigmoid is called softmax function which squashes the outputs of each unit to be between 0 and 1, and also divides the each output such that the sum of total outputs is equal to 1. The output is a categorical probability distribution, which gives the probability that any classes are true.

Instead of classic stochastic gradient descent optimation I used the Adam Optimizer. Adam Optimizer[3] computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients. To evaluate the models performance I used the Categorical Cross-Entropy loss function which captures the deviation based on predicted probability.

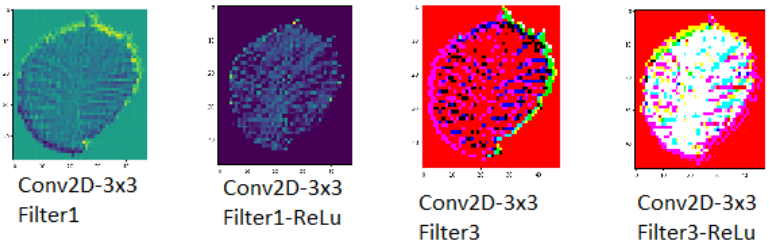

Figure 3. Visualization of activation maps

Figure3 is the visualization of couple of activation maps due to random initalization of weights. This gives some intuition of what filters can see after a convolution and basic features like corners that it tries to learn.

5.3. CNN with PCA Initialization of weights

CNN architecture with similar settings are used and only the weights of Convolutional Layer are changed through the help of custom initializer function. The architecture diagram for PCA initialization is explained in the figure4 below.

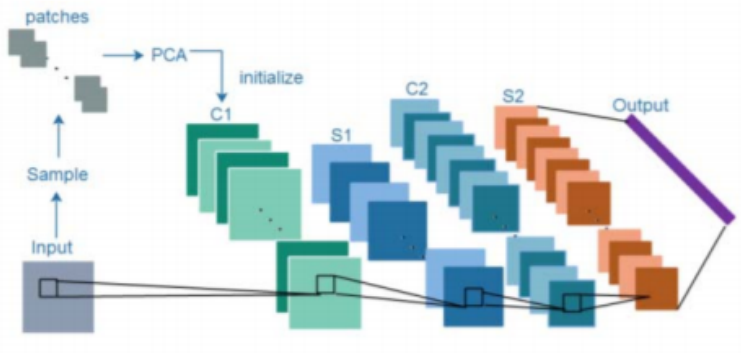

Figure 4. PCA initialization of weights to CNN

Random patches of images are generated from the input dataset according to the size of the convolution kernels followed by PCA. I performed PCA on 5x5 pixel image

patches, treating patches as points and pixels as dimensions. This gives 25 principal components, each with 25 element weight vector (which is an eigenvector of covariance matrix). Reshaping each PCA weight vector to a 5x5 vector gives basis image patches. Using all of the patches here account to 100 percent variance in the data. Reduced number of patches as explained by PCA varience graph (that captures more that 90 percent of the variance) can also be selected. I ran my experiments following both approaches which are essentially the number of input filters in Convolutional Layers and a slightly changed kernel size.

The visualization of image patches generated from the PCA approach can tell how these features are correlated to the input image. The color intensities are closer to the input images which gives a clue of how the weights initialization will be closer and better than the randomly initialized weights.

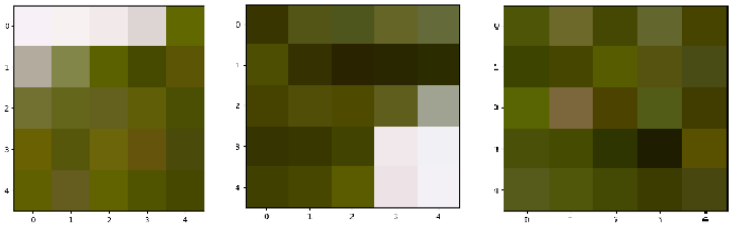

Figure 5. Images patches from PCA approach

5.4. SVM and NN

I also trained basic LinearSVM classifier and a simple Neural Network based on the input features alone to compare the performances. The LinearsVM used here is the multi-class version of Largest margin SVM with Linear kernel. The multi-class support is handled according to a one-vs-the-rest scheme. The Neural network architecture have 2 fully-connected layers $(128 * 15)$ with ReLu and softmax activation.

All experiments are coded with help of Python Keras Library utilizing the tensflow backend. Notable libraries used extensively are Numpy for vector operations and sci-kit learn library for applying algorithms like SVM, PCA etc.

6. Results

The table summarizes the results of different classifiers measured with prediction accuracy metric.

Model	Accuracy	Train Epochs
LinearSVM	0.7411	-
Basic Neural Network	0.7812	10
CNN	0.8734	10
CNN with PCA weights	0.8963	10

The results of first two classifiers were not so good which explains the complex nature of the problem and why a CNN type model is needed to approach this image classification task. Also, poor result of LinearSVM is an indicator of te non-linear nature of the

actual decision boundary. This is a result of more support vectors captured on the wrong side of margins yielded by the largest margin classifier. For CNN, different experiments were performed by changing the number of layers, input kernel size, including dropouts and the ones with best performances are reported here.

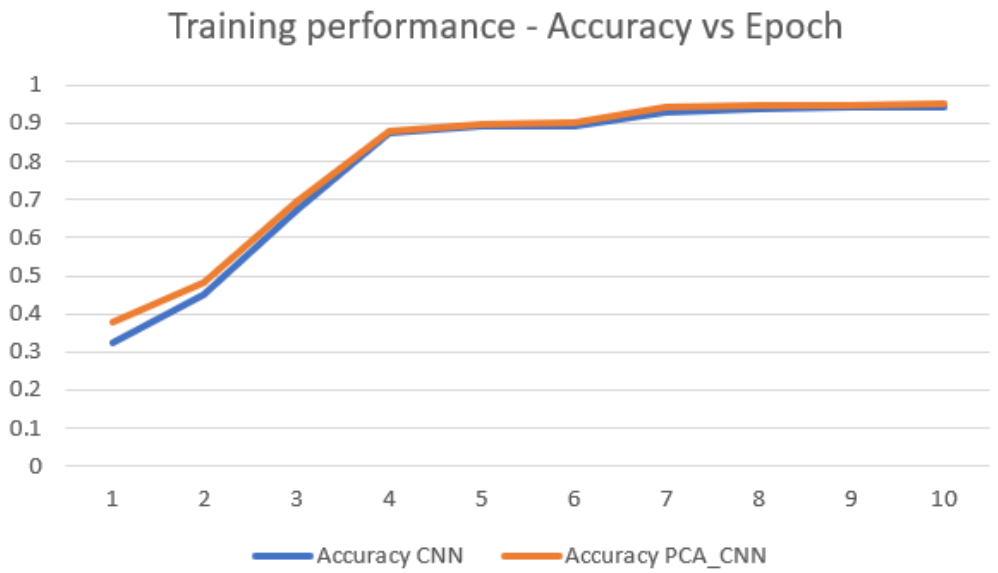

Figure 6. Training Performance

The actual performance comparison for CNN and CNN with PCA approach is done by plotting the training performance of these two models. As explained in the graph below, the PCA initialization approach does seem to perform better overall, but not by a large margin. The initial weights by PCA approach puts the network accuracy to a slightly better place and grows a bit faster in terms of convergence. But the convergence point (epoch) to the highest saturated accuracy is the same which is close to 95%. In both the cases, the prediction performance on test data is similar. The overfitting or model complexity which causes the generalization error gap does not actually seem to improve from the PCA approach, with this dataset and these experiments.

7. Conclusion and Future Work

It is evident from the results that CNNs are actually the better classifiers for this image classification task, again proving its powerful feature extraction capabilities. The PCA based approach through experiments and visualization actually revealed its nature of choosing features close to the input data distribution. Although the PCA based approach does seem to give notable performance in this work, re-visiting all other aspects of the classifier design could boost the prediction capabilities of the network. Also, the dataset in hand is very limited that the amount of images available for training could not contribute much to the weights initialization process or training performance as a whole. Although training with few samples is whole point of this work, the point that classic CNN alone can achieve good result could cloud the visible change in performance of PCA approach. I think similar design and experiments on a different dataset could yield visible

differences of between these approaches. Also, other techniques pertaining to CNN architectures to avoid overfitting like batch normalization, well-studied regularization could improve the performance both in this problem and similar problems.

References

[1] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *J. Mach. Learn. Res.*, 11:625–660, March 2010.

[2] T. J. Jassmann, R. Tashakkori, and R. M. Parry. Leaf classification utilizing a convolutional neural network. In *SoutheastCon 2015*, pages 1–3, April 2015.

[3] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc.

[5] 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.

[6] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[7] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[8] Oskar Söderkvist. Computer vision classification of leaves from swedish trees, 2001.

[9] Sebastian Thrun. Is learning the n-th thing any easier than learning the first? In *Proceedings of the 8th International Conference on Neural Information Processing Systems*, NIPS'95, pages 640–646, Cambridge, MA, USA, 1995. MIT Press.

[10] Raimar Wagner, Markus Thom, Roland Schweiger, Gunther Palm, and Albrecht Rothermel. Learning convolutional neural networks from few samples. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pages 1–7. IEEE, 2013.