CONVOLUTIONAL NEURAL NETWORKS

FOR CIFAR – 10 IMAGE CLASSIFICATION

Project Proposal

ECE 6254: Statistical Signal Processing
Spring 2016

Chia-Sheng Hsu (chiasheng@gatech.edu)

Jingting Yao (jingtingyaobecky@gatech.edu)

Rama Mythili Vadali (mythili.vadali@gatech.edu)

Yi-Chi Shao (ycshao@gatech.edu)

Summary

Objective: To apply and implement **convolutional neural networks (CNNs)** to classify CIFAR-10 images based on their themes/categories.

Motivation: Inspired by biological neural networks of brains, **artificial neural networks (ANNs)** are an important concept of machine learning. Convolutional neural networks (CNNs) are specialized ANNs models which exploit the spatial correlation and variabilities of 2D shapes to extract features, and therefore is one of the best among all the online and offline image classification networks as to the literature search so far [1]-[4]. Based on knowledge of ANNs and their layered architectures, this project is intended to build up understanding of the architecture of CNNs and obtain experience with training on a set of labeled images and consequently to **predict** the category of a random image.

Datasets: Popular datasets for image classification problem are CIFAR, MNIST and ImageNet. Limited by resources of personal laptops, especially insufficient RAM space and computational speed, the training and testing datasets chosen are **60,000 32x32 tiny color images in 10 classes** from **CIFAR-10** [5]-[7], with 6000 images per class. The training datasets consist of 50,000 images and testing 10,000 images. Both training and testing images are in random order. Training and testing datasets are mutually exclusive.

Methodology: The realization of classification consists of a series of modules, as illustrated in Fig. 1. The machine learning algorithm of this classification system is the CNN which is constructed by pre-processing, feature extraction and classification modules. The CNN architecture consists of a sequence of layers. Each layer could be a convolutional layer, pooling layer or a fully-connected layer. The initial plan of this project is to use and evaluate a proven/existing CNN architecture. Time permitting, further experiments with layer functionalities, order of feature extraction, other datasets and different/more class labels will be conducted. Performance **evaluation metrics** such as confusion matrix and ROC curve will be considered.

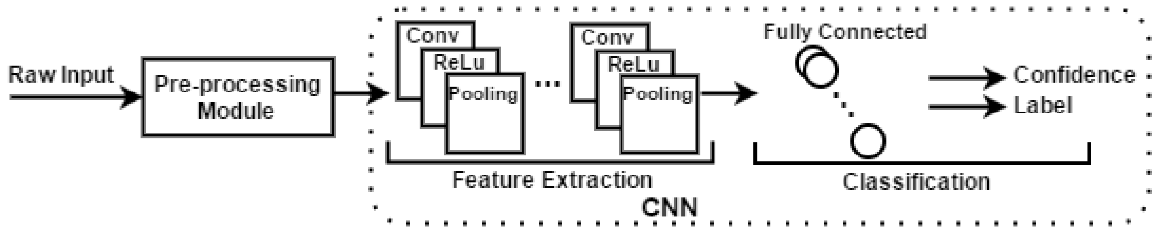

Figure 1[*]. Flow chart of classification methodology

*subject to change based on progress of the project

Further Explorations: More number of layers may result in better performance and thus is worth attempting if resources allowed. Furthermore, increasing the amount of training images narrows down the empirical risk. The number of image categories can also be extended to a greater scope by choosing CIFAR-100 dataset. The chosen CNN architecture can also be extended to other datasets having larger image sizes such as ImageNet dataset.

Detailed Project Description

1. Introduction

Convolutional neural networks (CNNs) are specialized ANNs models which exploit the spatial correlation and variabilities of 2D shapes to extract features, and therefore are one of the best among all the online and offline image classification networks as to the literature search so far [1]-[4]. Based on knowledge of ANNs and their layered architectures, this project is intended to build up understanding of the architecture of CNNs and obtain experience with training on a set of labeled images and consequently to predict the category of a random image.

CNNs[11] are similar to ordinary Neural Networks, but make the explicit assumption that there is spatial correlation in the input data. This makes CNN architectures well-suited learning algorithms for images and speech. Like ordinary Neural Networks, CNNs are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other. And they have a loss function like SVM or Softmax on the last (fully-connected) layer.

To take advantage of the fact that the input consists of images, each layer of CNN has neurons arranged in 3 dimensions: width, height, and depth (for RGB). The neurons in a layer will only be connected to a small region of the layer before it, instead of the neurons connected in a fully-connected manner. Every layer of CNN transforms the 3D input volume to a 3D output volume of neuron activations. The final output layer for CIFAR-10 would have dimensions 1x1x10, that represents a single vector of class scores arranged along the depth dimension.
Three main types of layers make up CNN architectures: **Convolutional layer, Pooling Layer, and Fully-Connected layer**. These layers are stacked up to form a full CNN architecture.

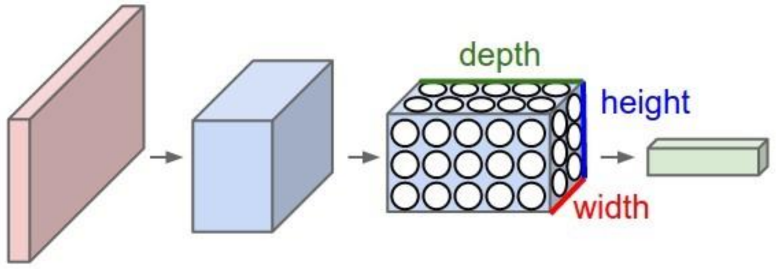

Figure 1: A CNN arranges its neurons in three dimensions, as visualized in the blue layers. The red input layer holds the image.

The **Convolutional Layer** is the core building block of a Convolutional Network, and its output volume can be interpreted as holding neurons arranged in a 3D volume. The output of convolution layer depends primarily on three hyperparameters: depth, stride and zero-padding. Depth controls the number of neurons in the Conv layer that connect to the same region of the input volume. Stride with which we allocate depth columns around the spatial dimensions (width and height). When the stride is 1, then we will allocate a new depth column of neurons to spatial positions only 1 spatial unit apart. This will lead to heavily overlapping receptive fields between the columns, and also to large output volumes. Conversely, if we use higher strides then the receptive fields will overlap less and the resulting output volume will have smaller dimensions spatially.

The description of CNNs in this section has been adapted primarily from [11].

2. Literature survey

In [5], **deep convolutional neural networks** are used to classify the image from ImageNet. On the test data, they achieved top-1 and top-5 error rates of 37.5% and 17.0% which is better than the previous works. The architecture contains eight layers which involves five convolutional layers and three fully-connected layers. Since the amount of computation is large, the author train on multiple GPUs to lessen the time. To reduce the overfitting effect, data augmentation and dropout skills are applied to the training data. They trained the models using gradient descent method and used equal learning rate for all layers. The results are compared with the best performance achieved during the ILSVRC-2010 competition as in [17] and the best published results since then as in [18]. In [19], the authors find that max-pooling can be replaced by a convolutional layers with increased stride, so they propose a new architecture consisting solely convolutional layers and the performance is competitive.

In the past years, there has been a lot of research conducted to improve the performance of this basic CNN architecture. Some work focuses on using complex activation function to improve classification (Goodfellow et al., 2013 [6]; Lin et al., 2014 [20]; Srivastava et al., 2013 [23]), while other work develops novel procedures to improve regularization (Zeiler & Fergus, 2013 [26]; Springenberg & Riedmiller, 2013 [27]; Wan et al., 2013 [28]). The success of CNNs (Krizhevsky et al. 2012 [5]) in the ImageNet challenge has inspired many different CNN architectural modifications for large scale object recognition.

The space of possible **activation functions** is explored in [7] to improve the accuracy of deep neural networks. An adaptive piecewise linear (APL) activation unit which is learned independently for each neuron using gradient descent is proposed, while traditional ANN applies fixed pre-defined non-linear activation functions to neurons. Compared to other nonlinear activation functions, specifically maxout [6] and network-in-network [12], APL involves less parameters and thus saves computational and modeling power. It is noted by [7] that the standard one-activation-function-fits-all approach could be suboptimal. Another approach to speed up the network is to maintain/reuse the feature extraction part of a previously trained network and only retrain the classification part on several datasets, as described in [13]-[16]. The full training takes considerably longer time than retraining, and results in comparable or worse accuracy depending on the dataset [13].

3. Methodology

The first objective of the project is to implement an existing CNN architecture on CIFAR-10 dataset and reproduce the results claimed. Based on literature survey and various proven architectures,the proven network is narrowed down to VGGNet. Following the successful implementation of VGGNet, we will experiment with the CNN architecture and hyperparameters based on literature survey.

3.1. Implementation of VGGNet

3.1.1. Preprocessing

Preprocessing by image normalization reduces the variability across images and features, and thereby improves the classification process. Before undertaking CNN implementation, the raw images from CIFAR-10 dataset will be preprocessed by centering. Centering can be achieved by subtracting the mean values of each pixel of the training set from each image. A typical image normalization approach is histogram equalization which adjusts image intensities to enhance contrast for each image.

3.1.2. VGGNet:

VGGNet was the network proposed by Karen Simonyan and Andrew Zisserman [1] and was the runner-up in ILSVRC 2014. The main contribution of VGGNet was in showing that the critical component for good performance of CNN is the depth of the network. It is currently one of the most preferred choice when extracting CNN features from images.
The layers that constitute the VGGNet are INPUT, CONV, RELU, POOL and FC. Various VGGNet configurations were proposed in the paper with number of CONV/FC layers ranging from 11 to 19. The best VGG configuration contains 16 layers and features an extremely homogeneous architecture that only performs 3x3 convolutions and 2x2 pooling from the beginning to the end. A downside of the VGGNet is that it is more expensive to evaluate and uses a lot more memory and parameters (140 million parameters for CIFAR-10).

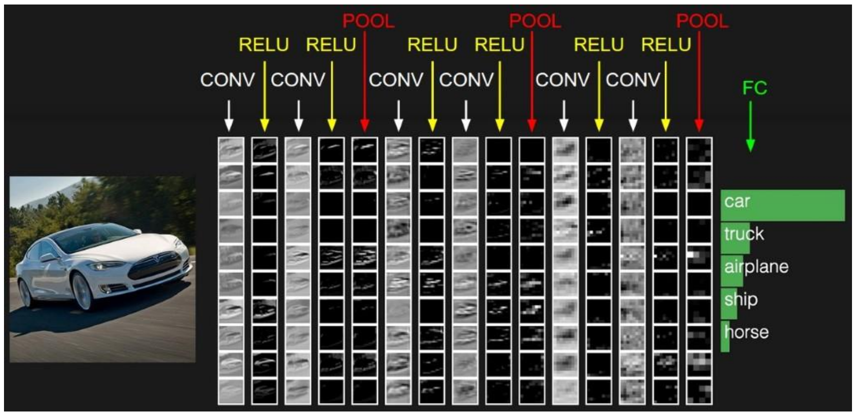

Figure 2: A tiny VGG Net architecture [11]

Here is an example to describe a tiny VGGNet configuration for CIFAR-10: [INPUT – CONV – RELU – POOL - FC]. These layers are illustrated in Fig. 2 and described below:

- INPUT [32x32x3] layer will hold the pixel values of the preprocessed image. For CIFAR-10 image this translates to an image of width 32, height 32, and with three color channels Red, Green, and Blue.
- CONV layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and the region they are connected to in the input volume. CONV layers of VGGNet perform 3x3 convolutions with stride 1 and pad 1 and will result in volume of [32x32x12].
- RELU layer will apply an elementwise activation function, such as the *max(0, x)* thresholding at zero. This leaves the size of the volume unchanged ([32x32x12]).
- POOL layer will perform a downsampling operation along the spatial dimensions (width, height). POOL layers of VGGNet perform 2x2 max pooling with stride 2 (and no padding), resulting in a volume of [16x16x12]

- FC (i.e. fully-connected) layer will compute the class scores, resulting in volume of size [1x1x10], where the 10 values correspond to the 10 categories of CIFAR-10.

3.2. Experiment with new techniques

Using the aforementioned VGGNet as a framework, improvements to the networks can be realized in terms of adjusting the activation functions, applying dropout after each fully-connected layer, and removing pooling layers but increasing the stride.

Activation function: In VGGNet framework, the activation functions are a single type of reset ramp function in ReLU. Other basic activation functions include logistic function, Gaussian, tangent hyperbolic function and maxout. Some recent proposed activation functions are network-in-network and adaptive piecewise linear units (APL). Different activation functions or their combinations may result in different accuracy and may even effect the convergence of the networks.

Dropout: Furthermore, dropout layers can be added after each fully connected layer. The term "dropout" refers to dropping out units (hidden and visible) in the CNN. Dropout prevents overfitting and provides a way of approximately combining exponentially many different neural network architectures efficiently.

Pooling: VGGNet uses max pooling while average pooling can be attempted by taking the average of the block. Based on [4], the pooling layer with pooling size equals to 3 and stride number equals to 2 could be replaced with a convolutional layer with corresponding stride and kernel size. And the number of output channels should equal to the input channels.

3.3. Algorithm evaluation metrics

The results will be evaluated via confusion matrix, ROC curve, or top-1 and top-5 method.
The classification error rate can be compared with the other proven CIFAR 10 algorithms (such as All-CNN [19], Maxout [5], Network in Network [20], Deeply Supervised [21], DropConnect [20], dasNet [22], AlexNet [5], Highway Network [24], and Fractional Max-Pooling [25]). For various experimented CNN configuration, the computational time and memory requirement for training will be computed and compared. This is an indication of scalability of the algorithm to larger datasets.

4. Further Exploration

More aspects of this project can be conducted in the future. For instance, increasing the amount of training images narrow down the empirical risk. The number of image categories can also be extended to a greater scope by choosing CIFAR-100 dataset. The chosen CNN architecture can also be extended to other datasets that have larger image sizes such as ImageNet dataset. At data preprocessing stage, discrete cosine transformation (DCT) can be applied to remove the high-frequency components and only use a few low-frequency components of DCT for representation, which could possibly remove the noise in the images, thereby improving performance of the learning algorithm. Lastly, object detection techniques can also be incorporated into our pipeline to help assign labels to images that have more than one object.

List of tasks and collaboration plan

Tasks	**Deadline**	**Importance**	**Leaders**
Brainstorm for project ideas and define our problem	March 28	Ideas are the foundation of everything	(1)(2)
Literature survey and background learning about CNN	April 3	The task needed to implement and design the pipeline	(3)(4)
Data collection (CIFAR-10) and preprocessing	April 3	The data set needs to be carefully chosen beacuse the performance highly depends on the input.	(1)(3)
Devise architecture of CNN	April 3	Main part of the project	(2)(4)
Estimate time required for implementation	April 6	Optional: just to get an idea of how much time we need for implementation	(1)(4)
Implement Algorithms	April 17	Main part of the project	(2)(3)
Performance evaluation	April 20	Compare with other literature	(3)(4)
Project report and poster	April 25	Analyze and interpret the outcome	(1)(2)

Team members: (1) Rama Mythili Vadali (2) Jingting Yao (3) Chia-Sheng Hsu (4) Yi-Chi Shao

Potential challenge

Computational challenges when more layers are introduced or the network is applied to a huge dataset with more categories.

References

[1] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).

[2] He, Kaiming, et al. "Deep Residual Learning for Image Recognition." *arXiv preprint arXiv:1512.03385* (2015).

[3] Szegedy, Christian. "Build a deeper understanding of images." *Google Research Blog*. N.p., 5 Sept. 2014.

[4] LeCun, Yann, et al. "Gradient-based learning applied to document recognition."*Proceedings of the IEEE* 86.11 (1998): 2278-2324.

[5] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*. 2012.

[6] Goodfellow, Ian J., Warde-Farley, David, Mirza, Mehdi, Courville, Aaron, and Bengio, Yoshua. Maxout networks. In ICML, 2013.

[7] Agostinelli, Forest, et al. "Learning activation functions to improve deep neural networks." *arXiv preprint arXiv:1412.6830* (2014).

[8] Benenson, Rodrigo. "What is the class of this image?." *Classification datasets results*. N.p., 2013.

[9] Krizhevsky, Alex, Vinod Nair, and Geoffrey Hinton. "The CIFAR-10 dataset." Learning Multiple Layers of Features from Tiny Images. N.p., 8 Apr. 2009.

[10] Krizhevsky, Alex. *cuda-convnet*. N.p., 18 July 2014. Retrieved from https://code.google.com/p/cuda-convnet/

[11] Karpathy, Andre, (2016). CS231n Convolutional Neural Networks for Visual Recognition. Retrieved from http://cs231n.github.io/convolutional-networks/.

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

[13] Hertel, Lars, et al. "Deep convolutional neural networks as generic feature extractors." *Neural Networks (IJCNN), 2015 International Joint Conference on*. IEEE, 2015.

[14] Razavian, Ali, et al. "CNN features off-the-shelf: an astounding baseline for recognition." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2014.

[15] Donahue, Jeff, et al. "Decaf: A deep convolutional activation feature for generic visual recognition." *arXiv preprint arXiv:1310.1531* (2013).

[16] Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014.

[17] A. Berg, J. Deng, and L. Fei-Fei. Large scale visual recognition challenge 2010. www.image-net.org/challenges. 2010.

[18] Sánchez, Jorge, and Florent Perronnin. "High-dimensional signature compression for large-scale image classification." *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011.

[19] Springenberg, Jost Tobias, et al. "Striving for simplicity: The all convolutional net." *arXiv preprint arXiv:1412.6806* (2014).

[20] Lin, Min, Chen, Qiang, and Yan, Shuicheng. Network in network. In ICLR: Conference Track, 2014.

[21] Lee, Chen-Yu, Xie, Saining, Gallagher, Patrick, Zhang, Zhengyou, and Tu, Zhuowen. Deeply su-pervised nets. In Deep Learning and Representation Learning Workshop, NIPS, 2014.

[22] Stollenga, Marijn F, Masci, Jonathan, Gomez, Faustino, and Schmidhuber, Ju¨rgen. Deep networks with internal selective attention through feedback connections. In NIPS, 2014.

[23] Srivastava, Nitish and Salakhutdinov, Ruslan. Discriminative transfer learning with tree-based priors. In NIPS. 2013.

[24] Srivastava, Rupesh Kumar, Greff, Klaus, and Schmidhuber, Jurgen. Training very deep networks. ¨ CoRR, abs/1507.06228, 2015.

[25] Graham, Benjamin. Fractional max-pooling. CoRR, abs/1412.6071, 2014. URL http://arxiv.org/abs/1412.6071.

[26] Zeiler, Matthew D. and Fergus, Rob. Stochastic pooling for regularization of deep convolutional neural networks. In ICLR, 2013.

[27] Springenberg, Jost Tobias and Riedmiller, Martin. Improving deep neural networks with probabilistic maxout units. In arXiv:1312.6116, also appeared at ICLR: Workshop Track, 2013.

[28] Wan, Li, Zeiler, Matthew D., Zhang, Sixin, LeCun, Yann, and Fergus, Rob. Regularization of neural networks using dropconnect. In International Conference on Machine Learning (ICML), 2013.