Convolutional Neural Networks Applied to Human Face Classification

Brian Cheung
Dept. of Electrical Engineering
The Cooper Union
New York, New York 10003
cheung4@cooper.edu

Abstract—Convolutional neural network models have covered a broad scope of computer vision applications, achieving competitive performance with minimal domain knowledge. In this work, we apply such a model to a task designed to deter automated systems. We trained a convolutional neural network to distinguish between images of human faces from computer generated avatars as part of the ICMLA 2012 Face Recognition Challenge. The network achieved a classification accuracy of 99% on the Avatar CAPTCHA dataset. Furthermore, we demonstrated the potential of utilizing support vector machines on the same problem and achieved equally competitive performance.

I. INTRODUCTION

The seminal work by LeCun et al. demonstrated the performance of convolutional neural network models (ConvNets) in handwritten digit classification, achieving impressive performance in the absense of explicit knowledge of the task [1], [2]. Notably, these models were found to be invariant to many different distractors including natural backgrounds, translations, rotations and even pose [3], [4]. Many of these favorable characteristics have led to the development of several variants of the ConvNet model with a broad range of applications [5], [6].

D’Souza et al. proposed a framework called Avatar CAPTCHA to distinguish human users from computer programs to prevent the abuse of computational resources by automated systems [7]. Users are presented with a set of images and asked to discriminate between human faces and computer generated avatars. Sample images from the dataset are shown in Figure 1. The underlying security of such a framework lies in the potential difficulty of an automated system to easily discriminate between these two classes, a task which they demonstrate is relatively simple for human users.

In this work, we demonstrate the effectiveness of ConvNets for this specific task as part of the ICMLA 2012 Face Recognition Challenge. In addition, we compare these results to a support vector machine (SVM) [8] which also attains competitive performance on the provided dataset. An attacker with an effective classifier could bypass the proposed human verification framework with a success rate substantially greater than brute force attacks.

II. DEEP LEARNING

ConvNets are one variant of a more general class of models in the field of deep learning. These models have a particular emphasis on many layered architectures. In a deep network, each layer contains nodes which computes a simple transformation of the input such as a weighted sum and/or activation function (e.g. sigmoid or hyperbolic tangent) [9]. While moving from one layer to the next, the input is transformed into progressively more abstract representations.

Bengio et al. differentiates various machine learning models in terms of different forms of representation-learning, the learning of representations or transformations of the input which make it easier to extract useful information (e.g. class label) [9]. Models which exploit local generalization learn a transformation which has parameters specific to local regions of the input space. Such models include decision trees, nearest-neighbors and support vector machines with a Gaussian kernel.

In contrast, deep learning models exploit the concept of distributed representations, where each parameter can be sensitive to a very broad range of the input space. This allows the learned transformation to be more expressive; able to properly represent many different input configurations. For example, a desireable representation for an object
recognition task would be invariant to varying degrees of rotation or scale. These two forms of representation learning can be combined to create a classifier where the distributed representation learned by a deep network is used to find an expressive representation (i.e., feature space). This in turn can be used as input for a model which learns local generalizations [9].

These transformations are often learned in an unsupervised manner where class labels are not available to the model. This is possible under the assumptions of the manifold hypothesis, which states that the probability distribution which generates the data lies on some low-dimensional manifold [10]. In addition, deep learning models can be trained in a semi-supervised manner where labels are only available for a subset of the training data. This is possible because the factors of the underlying distribution which generates the data are shared. In the context of transfer learning, only some of these factors need to be shared between the labeled and unlabeled data [9]. Beyond transfer learning, Raina et al. showed that performance in classification tasks could even be improved by training on samples of a distribution indirectly related to the classification task (self-taught learning) [11].

Deep learning is also related to well-known dimensionality reduction methods such as independent component analysis (ICA) and sparse coding. Le et al. defined a formal connection between the three by expressing the respective cost functions of each in a common format and described circumstances in which they become mathematically identical [12]. Such flexibility in defining the cost function of deep learning models has also led to many novel methods for imposing regularization constraints from other machine learning fields [13], [14].

III. CONVOLUTIONAL NEURAL NETWORKS

ConvNets have shown a great deal of promise in a wide variety of tasks from audio classification [15] to connectomics, a field which studies biological neural connectivity [16]. Weakly inspired by observations of neural processing in the visual cortex of the brain itself [17], [18], [19], these models are believed to learn an internal representation which progressively becomes more invariant to the classification task. Unlike other deep learning models which usually learn a distributed representation from the data, ConvNets also have an invariance property built-in which is particularly suitable for image classification tasks. Goodfellow et al. discovered that this hard-wired invariance allows ConvNets to acquire more invariance from layer to layer than its non-image-specific deep learning counterparts [20].

Being one of the first successful deep learning models, the early ConvNets have much in common with standard multilayer perceptrons. In contrast to other deep learning models, ConvNets can be trained in a completely supervised manner with the well-known backpropagation method [21]. These networks do not suffer from the vanishing (or exploding) gradient problem [22] which prevented many deep learning models from being successfully trained with backpropagation.

Lawrence et al. successfully applied ConvNets to a face recognition task where individuals in the dataset contained varying facial expressions [23]. Rather than using the face image as a direct input to the ConvNet, they constructed a hybrid model by combining a self-organizing-map [24] for preprocessing (dimensionality reduction) and a ConvNet for feature extraction and classification. At that time, their system achieved state-of-the-art classification performance and was significantly faster than the second best performing system [23].

IV. METHODS

We build a ConvNet in the Python programming language using Theano, a mathematical expression compiler tailored towards deep learning models⁠[1] [25]. In this work, the ConvNet is composed of alternating layers of convolution and subsampling as illustrated in Figure 2. This model contains six layers between the input image and the classification output. The convolutional layers (C1,C3) convolve the input with a learned kernel. The subsampling layers (S2,S4) perform a max-pooling operation which down-samples the data by partitioning the image into equally sized non-overlapping regions and outputs the maximum value in each region. After each max-pooling transformation, the output passes through a hyperbolic tangent function before becoming the input to the following layer. The next layer is a standard perceptron hidden layer which also uses a hyperbolic tangent function. The final output layer performs a sigmoidal operation.

We use backpropagation with stochastic gradient descent [21] to find the model parameters, \( \theta \), which locally minimizes the negative log-likelihood (cost) function, \( \ell(\theta, D) \), shown in Equation 5.

<sup>1</sup>Examples available at https://github.com/lisa-lab/DeepLearningTutorials
\[ \hat{y} = f(x, \theta) \] (1)

\[ P(Y = k|x, \theta) = \frac{e^{\hat{y}_{1k}}}{\sum_{i=1}^{N} e^{\hat{y}_{i}}} \] (2)

\[ D = \{ (x^{(i)}, y^{(i)}) : i = 1 \ldots N \} \] (3)

\[ \mathcal{L}(\theta, D) = \sum_{i} \log(P(Y = y^{(i)}|x^{(i)}, \theta)) \] (4)

\[ \ell(\theta, D) = -\mathcal{L}(\theta, D) \] (5)

In this work, we transform the output vector of the ConvNet model, \( \hat{y} = (\hat{y}_1, \hat{y}_2) \), using the softmax function [25] as illustrated in Equation 2. We then compute the log-likelihood function, \( \mathcal{L}(\theta, D) \), in Equation 4 over a given dataset \( D \) containing \( N \) samples.

To address overfitting, we apply an early-stopping criterion [22]. We divide the training set, \( D_{\text{train}} \), into two parts: an inner-training set, \( D_s \), and a validation set, \( D_v \):

\[ D_{\text{train}} = D_s \cup D_v \] (6)

\[ D_s = \{ (x_s^{(i)}, y_s^{(i)}) : i = 1 \ldots T_s \} \] (7)

\[ D_v = \{ (x_v^{(j)}, y_v^{(j)}) : j = 1 \ldots T_v \} \] (8)

Samples from \( D_s \) are used to adjust model parameters while samples from \( D_v \) are used to assess model performance. Because of the small dataset size, the resolution of classification accuracy was low and often identical between training iterations. To resolve this ambiguity, our implementation calculated the negative log-likelihood of the ConvNet on the validation set, \( \ell(\theta, D_v) \), to compare the performance between training iterations.

We use the implementation of a support vector machine available in the scikit-learn machine learning package [26] to provide performance comparisons against the ConvNet model. We use the radial basis function (Gaussian) kernel for the support vector machine with a kernel coefficient equal to the dimensionality of the input vector \( x \). For \( SVM - RBF \), the adjustable penalty parameter, \( C \) [8], of the support vector machine cost function is fixed to the default value 1.0. For \( SVM - RBF_{gs} \), we also perform a grid search [26] to optimize this parameter with the training dataset, \( D_s \).

V. EXPERIMENTS

Following standard practice, expected classification accuracy is calculated using 10-fold cross-validation [27], [28]. Samples of the avatar and human datasets are combined to create a single dataset of 100 samples. The order of the dataset array is defined randomly by shuffling the samples of avatars and humans. This ensures partitions of the dataset will share a similar distribution of avatars and humans. The dataset array is then partitioned into 10 subsets, each containing 10 samples. Of the 10 subsets, 9 are used to train the model and the remaining subset is used for testing. This is performed 10 times where each subset is used for testing once. Finally, the classification accuracy of each subset is averaged, resulting in the expected classification accuracy.

VI. RESULTS

Cross-validation performance results are shown in Table I. Both the performance of the ConvNet and \( SVM - RBF_{gs} \) models for each fold are relatively good with only one error occurring in fold 4. The expected classification accuracy of the two was 99%. Using the default penalty, \( C = 1.0 \), \( SVM - RBF \), performed noticeably worse with an expected accuracy of 94%.

VII. CONCLUSION

We achieve good performance when applying ConvNets to the Avatar CAPTCHA dataset. Such performance far exceeds brute force attack and would enable an attacker to bypass the Avatar CAPTCHA with far fewer expected attempts. With the addition of grid search, a support vector machine is also able to achieve competitive performance. But given the small size of the provided dataset, it is difficult to determine how similar the performance between these two models are. These preliminary findings merit further exploration of the capabilities of ConvNets as well as Gaussian support vector machines in visual tasks where class differences are subtle and not easily quantified.

ACKNOWLEDGMENT

The author would like to thank the organizers of the ICMLA Face Recognition Challenge and the authors of [7] for making the Avatar CAPTCHA dataset available.

REFERENCES


