SoSimple: Convolutional Neural Networks and Transfer Learning for Mathematical Symbol Classification

Archit Jaiswal [1], Nathan Grinalds[2], Humberto Alejandro Garcia[1], and Amit Kumar[1]
[1]Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL
[2]Department of Materials Science and Engineering, University of Florida, Gainesville, FL

Convolutional neural networks (CNNs) are rapidly emerging as a popular choice for image classification because of their ability to extract highly detailed representations of images. Here, we demonstrate that transfer learning of the VGG16 CNN architecture on the ImageNet data set can be applied to classification of 10 handwritten mathematical symbols generated from 100 students. By freezing the first 8 hidden layers of the pre-trained VGG16 model and allowing the next 8 hidden layers to adapt to our training data, we achieved $> 95\%$ classification accuracy on our validation set. It was discovered that the optimal learning rate for our model was 0.001 and that the stochastic gradient descent optimizer outperformed the Adam optimizer. From this study, we demonstrated that transfer learning is an effective method to deploy state-of-the-art CNNs on a carefully constructed set of images, however; some CNN to not generalize well to arbitrary datasets and can be computationally expensive.

I. Introduction

INNOVATIONS in machine-learning technology are strongly influencing the social and economic landscape of the modern world. Machine learning is helping enhance the performance of smart phones, web browsing, speech recognition, online commerce, drug discovery, and social networks by extracting patterns in data to make predictions [1]. Artificial neural networks (ANNs) are a popular choice for machine learning and information processing because its computational structure adapts based on the data fed into it (akin to biological neurons) [2-6]. ANNs have been identified as being particularly useful for classification problems like image recognition and sequential decision making. With the appropriate number of layers and neurons in each layer, standard ANNs are effective tools for processing sequential (temporal) information because of their recurring network architecture but are less adept at handling spatial data that would be used for classification tasks like image recognition. To overcome this shortcoming, convolution neural networks (CNNs) were developed which rely on convolutional layers that contain filters that help extract features from images. Layers that are close to the input obtain low-level features like edges and boundaries, while deeper layers learn more complex features like shadows and specific objects.

To motivate researchers to develop more effective CNN algorithms for image recognition and computer vision, the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) was launched in 2010 where competitors were tasked to develop an algorithm capable of classifying 1000 symbols from a dataset of 1 million images. The winners of the competition in 2014 developed a CNN architecture called VGG16 which was unique in its choice of small kernel sizes used to convolve images. By choosing a smaller kernel, the team was able to make their model deeper which enabled them to achieve 92.7% accuracy in classifying images belonging to 1000 classes [7]. Herein, we describe a CNN model that utilizes the VGG16 architecture that has been trained via transfer learning. The entire ImageNet dataset comprising 14 million images was used to train the VGG16 model which was then applied to the classroom dataset described below. Transfer learning bypasses the challenging and computationally expensive part of training neural networks by transferring a pre-trained model to an unknown dataset. Our approach was motivated by the recent success transfer learning and CNN has demonstrated for similar classification tasks [8].

II. Implementation

A. CNN Architecture

The VGG16 architecture begins with an input layer where image data is passed to with the resolution of choice (we chose 200x200 to balance resolution with memory requirements). From here, the image data is passed through 13 convolutional layers with pooling layers sandwiched every 2-3 layers. The data is then flattened, passed through 2 dense layers that are fully connected, and sent to the output layer.

The Relu activation function was assigned to all hidden layers so no negative values would be passed to

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 200, 200, 3)]	0
block1_conv1 (Conv2D)	(None, 200, 200, 64)	1792
block1_conv2 (Conv2D)	(None, 200, 200, 64)	36928
block1_pool (MaxPooling2D)	(None, 100, 100, 64)	0
block2_conv1 (Conv2D)	(None, 100, 100, 128)	73856
block2_conv2 (Conv2D)	(None, 100, 100, 128)	147584
block2_pool (MaxPooling2D)	(None, 50, 50, 128)	0
block3_conv1 (Conv2D)	(None, 50, 50, 256)	295168
block3_conv2 (Conv2D)	(None, 50, 50, 256)	590080
block3_conv3 (Conv2D)	(None, 50, 50, 256)	590080
block3_pool (MaxPooling2D)	(None, 25, 25, 256)	0
block4_conv1 (Conv2D)	(None, 25, 25, 512)	1180160
block4_conv2 (Conv2D)	(None, 25, 25, 512)	2359808
block4_conv3 (Conv2D)	(None, 25, 25, 512)	2359808
block4_pool (MaxPooling2D)	(None, 12, 12, 512)	0
block5_conv1 (Conv2D)	(None, 12, 12, 512)	2359808
block5_conv2 (Conv2D)	(None, 12, 12, 512)	2359808
block5_conv3 (Conv2D)	(None, 12, 12, 512)	2359808
block5_pool (MaxPooling2D)	(None, 6, 6, 512)	0
global_max_pooling2d (Global	(None, 512)	0
flatten (Flatten)	(None, 512)	0
dense (Dense)	(None, 1024)	525312
dense_1 (Dense)	(None, 11)	11275

Total params: 15,251,275
Trainable params: 15,251,275
Non-trainable params: 0

Fig. 1: Summary of model parameters.

a subsequent layer, and the Softmax activation function with 11 units was used for the output layers to assign a value between 0 and 1 to each class based on the probability the model determined. A summary of the model parameters is shown in Figure 1.

After compiling the model, the first 8 of its 16 layers were frozen so that half of the pre-trained weights obtained from the ImageNet training remained while the other (deeper) half could be adapted to our dataset of interest. We split our dataset into 70% training and 30% validation and subjected the model to the sparse categorical cross-entropy loss function. For all experiments, Nesterov's momentum (0.9) was used for gradient descent to ensure efficient convergence to a minima of our loss functions. Classification accuracy for the training and test sets was used as our performance metrics which we used to optimize the model's hyperparameters including the optimizer, learning rate, and batch size. The percent accuracy reported here is either the accuracy at the last epoch or the accuracy at the point where the validation accuracy no longer improves after 5 epochs. Our models were trained using the Keras API running on Tensorflow 2 to allow efficient operations on GPUs.

B. Data Collection and Processing

The dataset used to train our CNN model (hereby referred to as the classroom data set) was comprised of 9032 images falling into 10 classes of mathematical symbols: x, $\sqrt{}$, $+$, $-$, $=$, $\%$, ∂, $\prod$, π, $\sum$. All symbols were handwritten by students in Dr. Silva's Fall 2020 EEL-5840 class where a variety of fonts and handwriting styles were anticipated. The images were pre-processed by applying rotations to achieve proper orientations. Additionally, symbols that did not fall into one of the 10 images classes or were unrecognizable were assigned to an eleventh class for "unknowns". The images were then standardized to generate images of the same size and normalized by dividing the pixel values by 255 to obtain values ranging from 0 to 1.

III. Experiments

Prior to deciding to use transfer learning with the VGG16 architecture, our team experimented with other popular ANN models to determine the optimal model for our dataset. The VGG16 model was originally trained without transfer learning using the classroom data set which achieved a 90% accuracy on the validation set after some hyperparameter tuning. Resnet18, another popular CNN with 18 layers, was also deployed on the classroom dataset and achieved a 75% accuracy on the validation set. The InceptionV3 model with transfer learning was deployed on the classroom dataset and achieved a 54% accuracy on the validation set. Despite achieving 90% accuracy during our exploratory phase, we remained unsatisfied and decided to implement transfer learning with the VGG16 architecture which proved to be the most successful model. To determine the optimal hyperparameters for our model, we analyzed learning curves over 20 epochs after modifying batch sizes, number of frozen neurons, learning rate, and optimizer type. ***We achieved the highest accuracy (95.3%) in validation with the stochastic gradient descent (SGD) optimizer, a learning rate = 0.001, and batch size = 20)*** (Figure 2).

Using identical hyperparameters except changing the SGD optimizer to the Adam optimizer, the validation

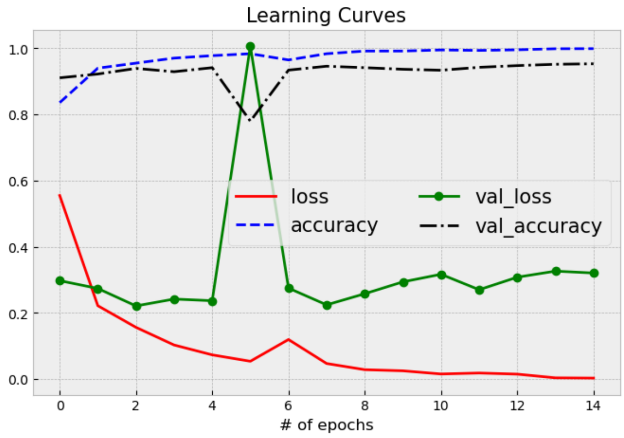

Fig. 2: Learning curve for the optimized model. Validation accuracy = 95.3%, learning rate = 0.001, optimizer = SGD, batch size = 20.

accuracy decreased to 86.0%. While we are unsure why the SGD optimizer outperforms the Adam optimizer in this scenario, it is known that the Adam optimizer tends to perform well in training but does not always generalize well [9].

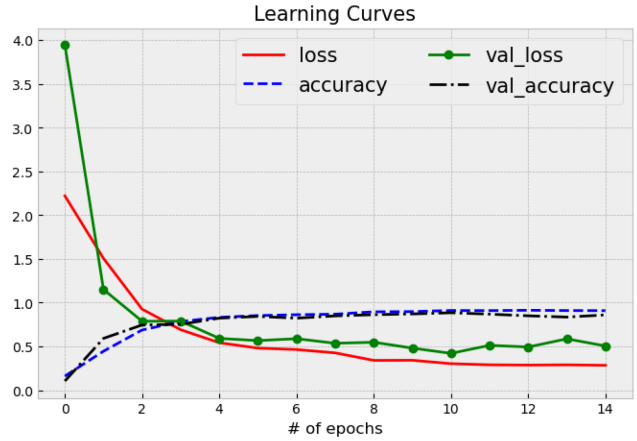

Fig. 3: Learning curve 2 for model with Adam optimizer. Validation accuracy = 86.0%, learning rate = 0.001, optimizer = Adam, batch size = 20.

IV. Conclusion and Future Directions

Although we achieved 95.3% classification accuracy on our dataset, future steps would involve a more formal series of cross-validation to optimize all of the hyperparameters of our dataset. Further, to improve the performance of our model (i.e., accuracy, run time, etc.) we could have also experimented with implementing batch normalization, regularization, and/or drop out into our model.

The next step to take involves identifying several mathematical symbols in an equation. Our final model proved to be successful in correctly identifying single symbols, but would not be able to pick out several symbols in one image. In order for this to be done, we would have to use a different model than the VGG16 model. The VGG16 model that we used is an image classification model that only works given images with one object. Since an equation consists of multiple objects or symbols, a model that can utilize object detection and image segmentation is a necessity. Image segmentation is the process of defining object boundaries within an image [10]. An example of a model architecture that is useful for object detection and image segmentation is the Fully Convolutional Network (FCN) model [10].

Given the success of transfer learning and CNNs to the classroom dataset, we are excited to implement similar CNN models into our own research projects after gaining familiarity with the techniques implemented into this project.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature,* vol. 521, no. 7553, pp. 436–444, May 2015,

[2] L. O. Chua and T. Roska, "The CNN paradigm," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications,* vol. 40, no. 3, pp. 147–156, Mar. 1993.

[3] Y. Sun, B. Xue, M. Zhang, G. G. Yen, and J. Lv, "Automatically Designing CNN Architectures Using the Genetic Algorithm for Image Classification," *IEEE Transactions on Cybernetics,* vol. 50, no. 9, pp. 3840–3854, Sep. 2020.

[4] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising," *IEEE Transactions on Image Processing,* vol. 26, no. 7, pp. 3142–3155, Jul. 2017.

[5] W. Zhang, K. Ma, J. Yan, D. Deng, and Z. Wang, "Blind Image Quality Assessment Using a Deep Bilinear Convolutional Neural Network," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 30, no. 1, pp. 36–47, Jan. 2020.

[6] Y. Li, J. Zeng, S. Shan, and X. Chen, "Occlusion Aware Facial Expression Recognition Using CNN With Attention Mechanism," *IEEE Transactions on Image Processing,* vol. 28, no. 5, pp. 2439–2450, May 2019.

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556,* 2014.

[8] M. Hussain, J. J. Bird, and D. R. Faria, "A Study on CNN Transfer Learning for Image Classification," in Advances in Computational Intelligence Systems, Cham, 2019, pp. 191–202.

[9] N. S. Keskar and R. Socher, "Improving generalization performance by switching from adam to sgd," *arXiv preprint arXiv:1712.07628,* 2017.

[10] T. Hassanzadeh, L. G. C. Hamey and K. Ho-Shon, "Convolutional Neural Networks for Prostate Magnetic Resonance Image Segmentation," in *IEEE Access,* vol. 7, pp. 36748-36760, 2019.