Extracting Image Features Through Deep Learning

Malihe Ghasemzade
Department of Metallurgical Engineering, Islamic Azad University, Karaj, Iran.
Email: gh_mavad@gmail.com

Received: March 2020 Revised: May 2020 Accepted: July 2020

ABSTRACT:
The purpose of this study is to identify images with deep learning with the least error. In machine learning projects, the basis of the work is extracting features from raw data. Finally, we differentiate different features through classifiers. In the present project, images with dimensions of 224*224 are applied to the network. Most networks use color images, which have 3 channels, the final dimensions of which are 3*224*224. We used the vgg19 network to extract the feature from the image with the highest accuracy. To increase the speed of weight correction operations, batch_size = 30 is considered. 70% of the images were used for network training, 20% for validation and 10% of the data for network testing and evaluation. The speed and accuracy of this project is high.

KEYWORDS: Deep Convolutional Network Learning, Supervised learning, Deep Learning.

1. INTRODUCTION

Deep learning is the construction of machine learning models that are used for hierarchical demonstrative learning of data. Deep neural networks are a general term for a set of neural networks with a multi-layered architecture that show how neural networks with a large number of layers can be successful in creating the representational structures needed for deep learning. In fact, there are various types of deep neural networks, including: autoencoder networks, convolutional neural networks and recurrent neural networks [1]. Different combinations of these networks can also be used according to the type of input space in the problem under study [2], [1]. These networks are formed by combining several nonlinear transformations with the aim of achieving more abstraction and ultimately useful representations of data. [3], [1]. The more layers in the neural network, the more complex the optimization problem. [1], [4-7]

Deep convolutional networks, like other deep learning methods, are composed of many layers, each of which is called a convolutional layer. Convolutional action in signal processing is defined on two signals. Convolutional neural networks are very similar to artificial neural networks. These types of networks consist of neurons with learnable (adjustable) weights and biases. Each neuron receives a number of inputs and then calculates the product of the weights multiplied by the inputs, and finally presents a result using a nonlinear conversion (activation) function. The whole network still provides a derivative score function. These types of networks still have a loss function like Softmax in the last layer, which is completely related, and all the points about normal neural networks are true here as well. Neural networks receive an input in the form of a vector and then pass it through a number of hidden layers. Finally, an output that results from the processing of hidden layers appears in the output layer of the network. Each hidden layer is made up of a number of neurons that connect to all the neurons in the previous layer. The neurons in each layer act independently and have no connection to each other. The last layer is completely connected to the output layer and usually plays the role of representing the score of each class. Convolutional neural networks took advantage of the fact that the input contains images and limited the network architecture in a reasonable way. In particular, unlike a typical neural network, the layers of a convolutional neural network include neurons arranged in three dimensions: width, height, and depth. The word depth here refers to the third dimension of an activator mass and does not mean the depth of a complete neural network, which means the number of layers in it. Instead of connecting to all the neurons in the previous layer, each neuron in each layer is connected to only a small area of the previous layer. The final output layer for images will be as the number of categories, because as we reach the end of the convolutional network architecture, we reduce the image size so that at the end, our complete input image is reduced to a vector containing category points (classes) and we will be faced with a vector that contains the points of each category. These scores are arranged along the depth dimension [8].

2. THE SUGGESTED METHOD

2.1. Deep Learning

In this project, we intend to differentiate the images. This project is based on deep learning. Deep learning is a sub-branch of machine learning. This method extracts the feature hierarchically from different layers through nonlinear functions. The input of each layer is the output of the previous layer and its training can be with or without a supervisor. In fact, the hidden monolayer in the neural network has been replaced by a large number of deep layers. The input layer in image processing projects is often raw images.

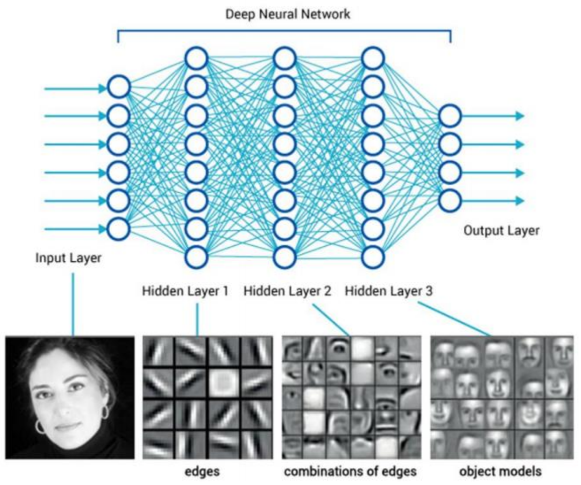

Fig. 1. A model of a deep neural network.

In machine learning projects, the basis of feature extraction is from raw data. For example, in image processing projects, we extract features from images stored from the camera through which we can distinguish images from each other. Features are smaller in size than the raw images, and in the end we differentiate the different features through the classifier. [9] In classification-based projects, the number of classes is mentioned. For example, in the present project, we have two classes. The features extracted from the face image are given to the classifier and the classifier consists of a combination of a series of linear and nonlinear functions, which in this project gives zero and one answer.

The number of classes in deep learning projects determines the size of the output layer, which in the present project, the size of the output layer is 2. The input layer is also determined by the network architecture. In the present project, 222x222 images are applied to the network. Most of the networks use color images containing 3 channels, the final dimensions are 222*222*3. So far, the input and output layers have been identified, what deep learning is supposed to do is to extract features from the image, and these features must be extracted in such a way that the classifier can separate the samples with the highest accuracy. We used the vgg19 network to do this, the reason for naming the deep network is that this feature extraction section uses a large number of layers to be able to extract our desired feature. Layers have different types that different networks are created by combining these layers in different states. As mentioned before, we are going to use the supervised network, each layer of the network is made up of a total of neurons, and the neurons contain weight and bias values. A round that uses all the training images is called an epoch. We may need hundreds of epochs to train the network, and with each epoch the accuracy of the model improves. The accuracy of the network is evaluated with a value called loss, because the classifier may misdiagnose a number of samples during the training operation. To estimate the amount of network error, we use a function called loss function. Low value for this function means that there is a small difference between the answer of the model and the desired answer, so the accuracy of the model is high and vice versa, if the loss function value is high, the accuracy is low and we must continue training. We terminate the network training in 2 cases, one is as the loss amount becomes so small that it is appropriate for us, or second when we tie the network to perform n-epoch training operations.

As seen above, there must be an optimization method to correct the weights by observing the amount of loss in each step. In the old methods, the correction operation was performed at the end of each epoch, in which the weight correction operation was performed slowly. That is, it took a long time to reduce the loss of our model, so to increase the speed of training after a smaller number of samples, the correction operation was performed. This number is called batch_size, which if the training data is assumed to be 2222 samples, taking into account 30=batch_size, means that after applying 32 samples to the network, the correction operation will be performed. In the old methods, after applying the whole data (2222 samples), correction operations were performed. SGD method is also used to update weights.

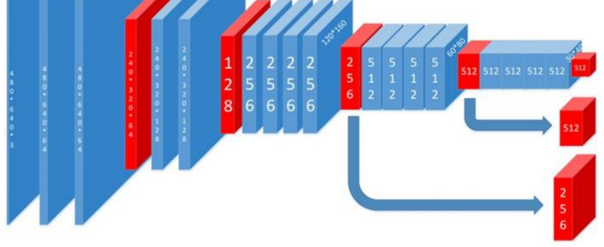

Fig. 2. Vgg19 Network architecture.

The basis of the vgg19 network is based on convolution and max-pooling, and as you can see in the architecture of this network, the convolution layers are shown in blue and the max-pooling layers are in red.

2.2. Convolution Operation

A kernel is a two-dimensional or one-dimensional array that has a set of coefficients such that the kernel has a central point called the anchor point. convolution in short means the action that takes place between any point of the image with the kernel. Now if we want to perform the convolution operation manually, we scroll the image from left to right and up and down. Suppose we use a 5 x 5 kernel and the current coordinate point (25,25) is in the image. Now we need to select an area of the image that is the same size as the kernel and multiply the elements in the kernel peer to peer and add all the values together and divide the result by the total kernel values. This area is from the coordinates (23,23) to the coordinates (22,22). Now we put the obtained value in the final image in the position (25,25). This operation is done for all the pixels. In fact, network learns local features in the image through the convolution layer.

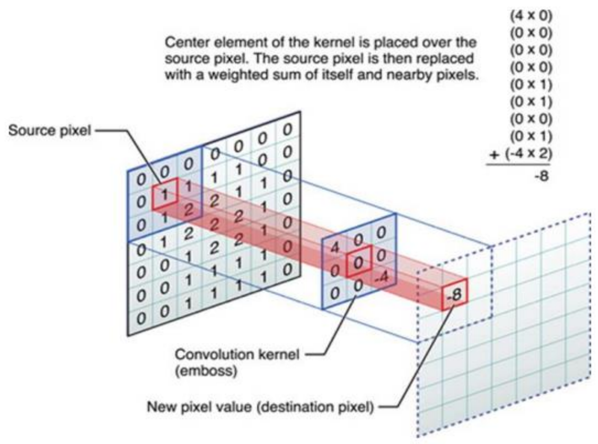
Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.
(4 x 0)
(0 x 0)
(0 x 0)
(0 x 0)
(0 x 1)
(0 x 1)
(0 x 0)
(0 x 1)
+ (-4 x 2)
-8
Source pixel
Convolution kernel (emboss)
New pixel value (destination pixel)

Fig. 3. Vgg19 Network architecture.

Max-pooling: Since the size of the initial layer of the network is large, we must use a method that reduces the dimensions of the layers along the network. If you look at the architecture of the vgg19, after applying each max-pooling layer, the dimensions of the layers become smaller.

2.3. Dataset

The UTKFace dataset contains 23,700 images. 70% of the images were used for network training, 20% for validation and 10% of the data for network testing and evaluation.

2.4. Modeling

VGG19 architecture and TensorFlow framework have been used to model the data. Weight training and weight correction continued for up to 120 epochs. To correct the weights, the SGD method with learning rate of e-51 has been used and the batch size has been considered 30.

2.5. Tensor Flow

TensorFlow is a powerful open source software library for numerical computing, designed and developed specifically for large-scale machine learning. Its basic principle is simple: first in Python, a diagram of the calculations to be performed is defined (for example, Figure 4), and then the TensorFlow takes that diagram and executes it using the optimized C ++ code. [9]

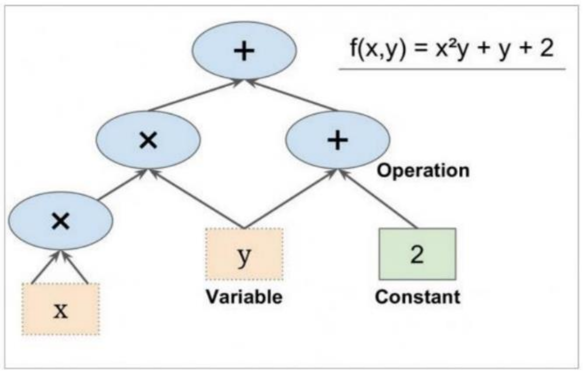

Fig. 4. Diagram of calculations.

Most importantly, the diagram can be broken into several pieces and run in parallel on multiple CPUs or GPUs. Tensorflow also supports distributed computing so that you can train huge neural networks on very large training packages in a reasonable amount of time by dividing the computations into hundreds of servers.

TensorFlow can train a network with millions of parameters on a training set consisting of billions of samples, each with millions of features, since TensorFlow was developed by the Google Brain team and uses the power of many large-scale Google services, such as Google Cloud Speech, Google Search and Google Photos. [9]

2.6. Tensor Flow in Batch Size

In each step of learning, a number of samples is taught and then the network parameters are set. This number of samples is called batchsize. Of course, it is important to choose the batchsize properly to not to be neither too big nor too small. On one hand, if the batchsize is equal to the total training samples, the gradient becomes more stable and the network converges slowly, and on the other hand, if the batchsize is selected too small, the gradient becomes unstable, which in turn forces us to reduce the learning rate.

2.7. Pooling Layer

The main purpose of the pooling layer is to subsample the input image to reduce the computational load, memory and number of parameters (reduction of overfitting risk). Reducing the size of the input image also makes the neural network less sensitive to image movement (independent of position).

Just like the convolution layer, each neuron in the pooling layer is connected to the output of a limited number of neurons in the previous layer. In fact, these neurons are located inside a small rectangular receptive field.

For the pooling layer, several items must be specified, including size, stride and padding type. Pooling layer neurons are weightless; in fact, they perform aggregation operation using an aggregation function such as maximum and mean.

Figure 5 shows the maximum type of pooling layer. In fact, this type of pooling layer is common in deep learning. In this example, the kernel size is 2*2 and the stride is 2 and we do not have padding. The maximum value in each kernel is sent to the output and other inputs are ignored.

As you can see, this type of layer is very subtractive. With a kernel size of 2x2 and a step size of 2, the size of the output image was halved in both directions. Simply put, 25% of the image was discarded.

In this case, the pooling layer acts independently on each channel, so it can be concluded that the input and output depths are equal. Of course there is also another case, when we do not intend to change the image dimensions and in fact we intend to perform the aggregation operation on the channels, for example, the mean of RGB values in 3 channels.

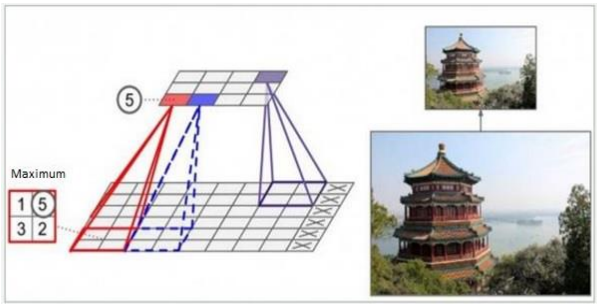

Fig. 5. Pooling layer.

2.8. Illustration

As you can see in Figure 6, the model prepared after 120 epochs has an accuracy of 22% in the network training phase.

As you can see in figure 7, the loss has been reduced to 0.18. As can be seen in Figure 8, the accuracy in the validation step is very fluctuating and has an accuracy of 92%.

As can be seen in Figure 9, the loss in the validation step is 0.08.

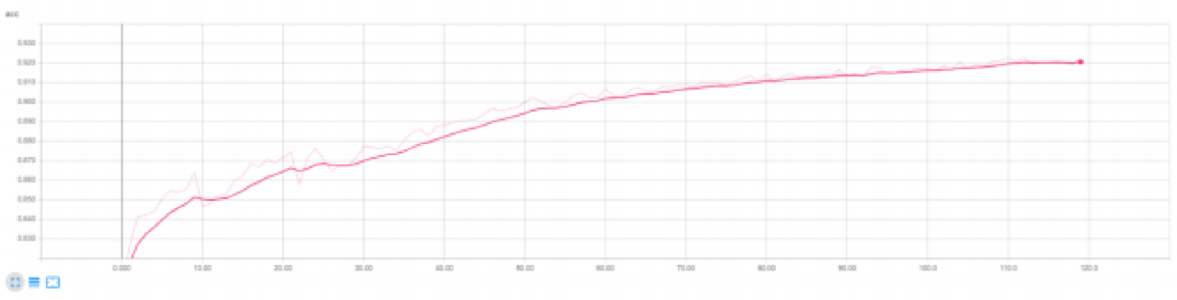

Fig. 6. The model prepared after 120 epochs with an accuracy of 22% in the network training phase.

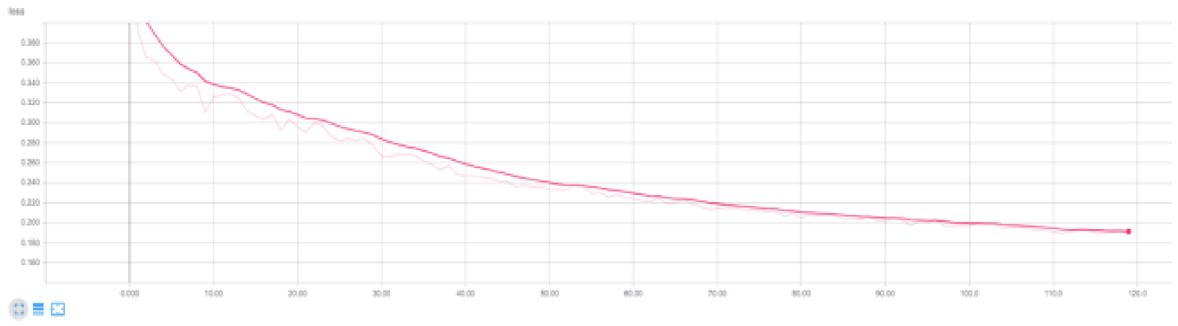

Fig. 7. Loss reduction up to 0.18.

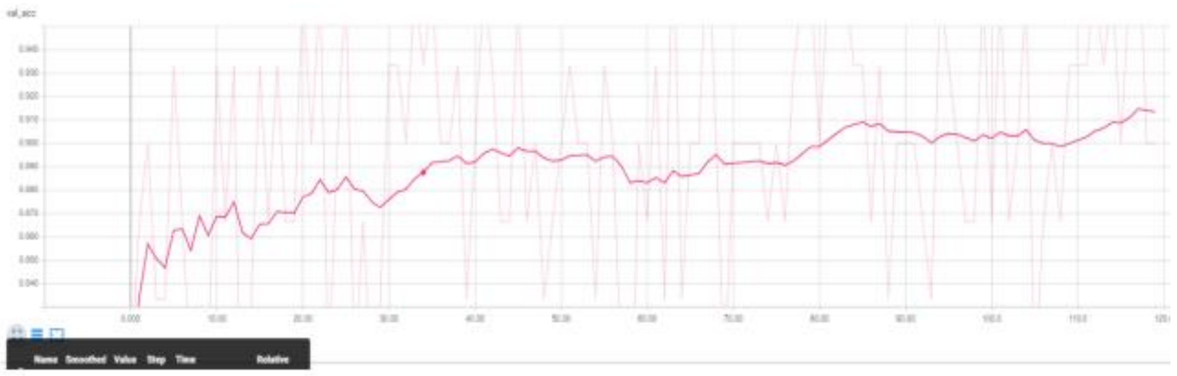

Fig. 8. Accuracy at the validation stage is 92%.

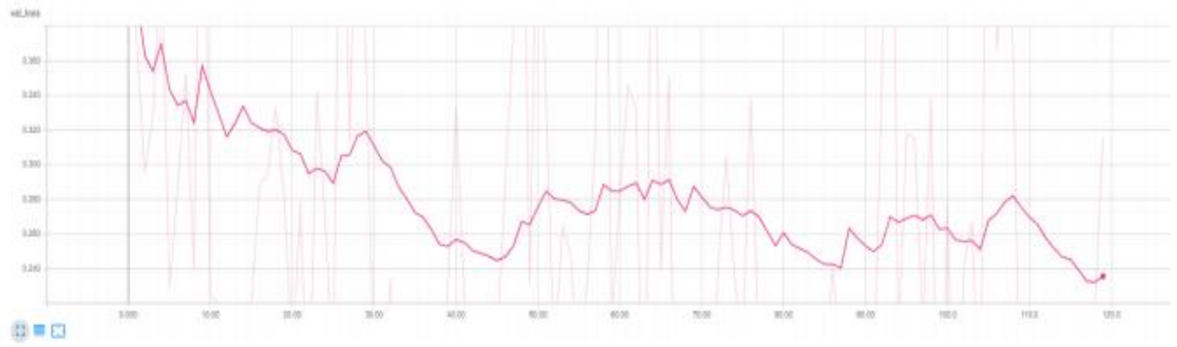

Fig. 9. Loss in the validation step is 0.08.

3. CONCLUSION

The project was based on deep learning. In machine learning projects, the basis of the work is extracting features from raw data. Finally, we differentiate different features through classifiers. In projects based on classification, the number of classes is considered. The classifier consists of a combination of a series of linear and nonlinear functions, which in this project gives zero and one answer. In the present project, the output layer size is 2. In the present project, images with dimensions of 224*224 are applied to the network. Most networks use color images, which have 3 channels and the final dimensions of which are 224*224*3. We used the vgg19 network to extract the feature from the image with the highest accuracy. We also used the supervised network. To estimate the amount of network error, we used a function called loss function. To increase the speed of weight correction operations, batch_size = 30 is considered, i.e. after applying 30 samples to the network, the correction operation was performed. In the old methods, after applying the whole data (2000 samples), correction operation was performed. The SGD method was also used to update the weights. 70% of the images were used for network training, 20% for validation and 10% of the data for network testing and evaluation. As shown in the results, the accuracy of our method was high.

REFERENCES

[1] Farinaz Alamian Harandi, Vali Darhami, **"Extraction of features from depth data using deep learning method for supervised control of a wheeled robot"**, *Journal of Control*, Vol. 11, No. 4, Winter 2017, pp. 24-13.

[2] Courville, I. G. a. Y. B. a. A., **"Deep Learning"**, *MIT Press*, 2016.

[3] Bengio, Y., Courville, A., & Vincent, P., "**Representation learning: A review and new perspectives**". *IEEE transactions on pattern analysis and machine intelligence*, Vol. 35(8), pp. 1798-1828, 2013.

[4] Bengio, Y., "**Learning deep architectures for AI**". *Foundations and trends® in Machine Learning, 2(1)*, pp. 1-127, 2009.

[5] Hinton, G. E., & Salakhutdinov, R. R., "**Reducing the dimensionality of data with neural networks**". *Science*, Vol. 313(5786), pp. 504-507, 2006.

[6] Lee, H., Grosse, R., Ranganath, R., & Ng, A. Y., "**Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations**", *Proceedings of the 26th annual international conference on machine learning*, 2009.

[7] Liu, J. N., Hu, Y., You, J. J., & Chan, P. W., "**Deep neural network based feature representation for weather forecasting**", *Proceedings on the International Conference on Artificial Intelligence* (ICAI), 2014.

[8] Mehdi Jamaseb Khalari,V. Derhami, "**Identify hand mode in video with deep learning**", *Sixth Joint Congress of Fuzzy Systems and Ho Shamand Iran* March 2010.

[9] http://www.7khatcode.com.