Dogs and Cats Binary Images Classification With Convolutional Network

Zeyuan Zhu, Haoyuan Qin

Northeastern University

CSYE7370 Deep Learning and Reinforcement Learning in Game Engineering

December 16, 2022

Abstract

Deep learning has become one of the most important breakthroughs in artificial intelligence over the past decade. It contains a variety of methods, including neural networks, hierarchical probabilistic models, and many specific unsupervised and supervised feature-learning algorithms.

In this project, we built a module that could recognize the image of dogs & cats by using Keras as the deep learning framework. The dataset from Kaggle contains 25000 pictures (12500 cats & 12500 dogs) training sets are created based on these pictures. Firstly, training and validate the network by original pictures 30 epochs. The validation accuracy is 75%, and the training accuracy is close to 100%. The reason for this is the module was overfitting these data. Then we modify train pictures by using data augmentation which changed these pictures randomly to make sure that our module will not receive duplicated pictures. And the validation accuracy after data augmentation is up to 80% and training accuracy is also closing to 80% after 100 epochs.

1 Introduction

"Deep Learning – which has emerged as an effective tool for analyzing big data – uses complex algorithms and artificial neural networks to train machines and computers so that they can learn from experience, classify and recognize data and images just like a human brain does." [1] Convolutional Neural Networks, also known as CNNs, isa subset of artificial neural networks being used in deep learning and are frequently employed for object and image recognition and classification. Thus, Deep Learning uses a CNN to identify items in an image. Due to their significant contribution to these rapidly developing and expanding fields, CNNs are widely used in deep learning. In this project, will build a cat/dog image classification system based on the CNN network.

1.1 Image classification

Image classification is the process of categorizing and labeling groups of pixels or vectors within an image based on specific rules.[2] The classification method can

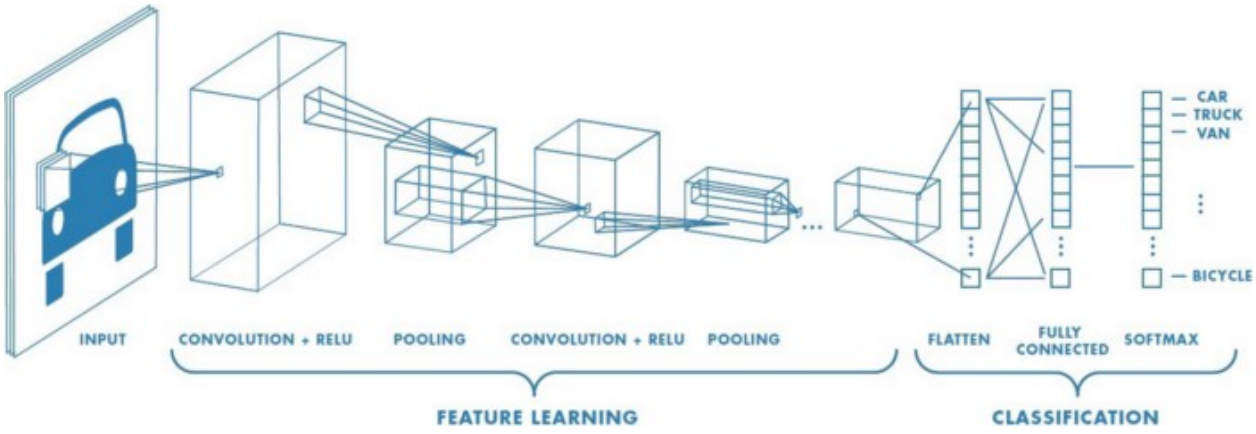
be used in different image features. ‘supervised’ and ‘unsupervised’ are the two most general methods.[3]

In supervised classification, having a sampling of the features. Every collection of features is categorized into a class that contains each class serves as a training sample. Multivariate statistics are computed on your training samples once you've determined who they are in order to establish the relationships both inside and between the classes. A signature file contains the statistics.

In unsupervised classification, you do not know what features are at any specified location, but you want to aggregate each of the locations into one of a specified number of groups or clusters. The multivariate statistics computed on the input are what define which class or cluster each location will be assigned. Based on the values for each cell within the clusters, each cluster is statistically distinct from the others.

In this project, we train a binary classification discriminator of cat and dog images using supervised Deep Learning Algorithms -- Convolutional Neural Network (CNN).

1.2 CNN model



Convolutional neural networks are composed of multiple layers of artificial neurons. Artificial neurons are mathematical functions that compute the weighted sum of several inputs and output an activation value, roughly imitating their biological counterparts. Each layer of a ConvNet creates several activation functions that are passed on to the following layer when an image is entered.

Typically, the first layer extracts fundamental features like edges that run horizontally or diagonally. The following layer receives this output and detects more intricate features like corners or multiple edges. The network may recognize increasingly more complex elements, including objects, faces, etc., as we go further into it.[4]

Based on the activation map of the final convolution layer, the classification layer outputs a set of confidence scores (values between 0 and 1) that specify how likely the image is to belong to a "class."

In this project, we built a ConvNet using Keras framework to detect cats and dogs, the output of the final layer shows the possibility of the image is a cat or a dog.

2 Data set

The dataset contains 25,000 images of dogs and cats. (12500 cats & 12500 dogs).[5] The image was collected from the Petfinder who is the world's largest site devoted to finding homes for homeless pets. They've provided over three million images of cats and dogs, manually classified by people at thousands of animal shelters across the United States. In this project, we built a discriminator model based on a subset of Petfinder data.

3 The Architecture

Our model is a sequential model that appropriates for a plain stack of layers where each layer has exactly one input tensor and one output tensor. Our model includes 3 convolutional layers and 3 pooling layers. The filter kernel size in each convolutional layer is 3x3 and the max pooling size is 2x2.

The depth of the feature map in the network is gradually increasing (from 32 to 128), and the size of the feature map is gradually decreasing (from 150-150 to 7-7) Increased depth means the original image is more complex and requires more filters Size reduction means more convolution and pooling layers are constantly compressing and abstracting the image.

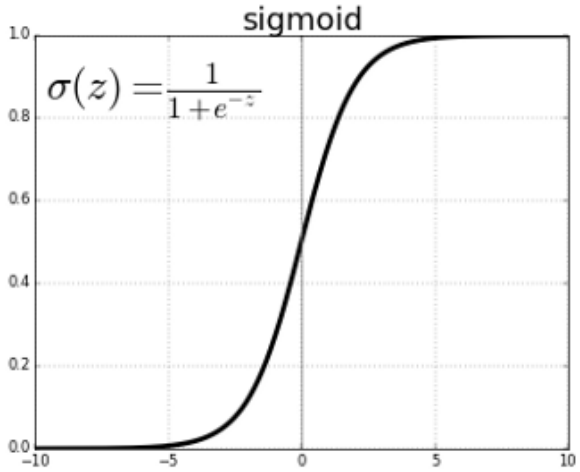
(a)Convolutional layer to extract features from 2D image with Relu activation function
(b)Max Pooling layer to down sample the image features
(c)Flattening layer to flatten data from 2D array to 1D array

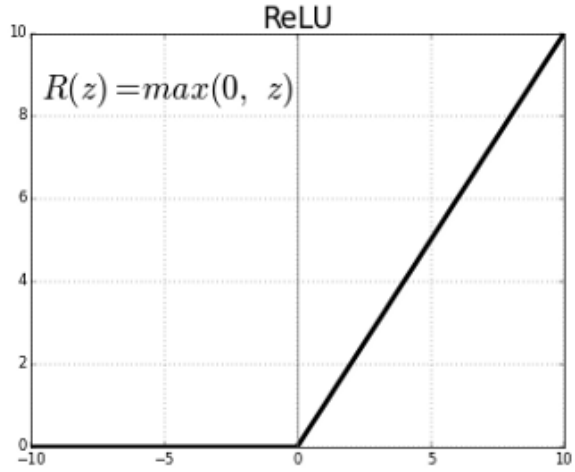
(d)Dense layer to capture more information
(e)Dense layer for output with sigmoid activation function

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d (MaxPooling2D)	(None, 74, 74, 32)	0
conv2d_1 (Conv2D)	(None, 72, 72, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 36, 36, 64)	0
conv2d_2 (Conv2D)	(None, 34, 34, 128)	73856
max_pooling2d_2 (MaxPooling2D)	(None, 17, 17, 128)	0
conv2d_3 (Conv2D)	(None, 15, 15, 128)	147584
max_pooling2d_3 (MaxPooling2D)	(None, 7, 7, 128)	0
flatten (Flatten)	(None, 6272)	0
dense (Dense)	(None, 512)	3211776
dense_1 (Dense)	(None, 1)	513

3.1 Activation function

In the convolutional layer, we use ReLU as activation function. ReLU is a non-linear function or piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. Those features make the train faster and reduced the likelihood of vanishing gradient. Because the ReLU's derivative is 1 for values larger than zero. multiplying 1 by itself several times still gives 1, The negative component of the ReLU function cannot be discriminated against because it is 0. As a result, negative values' derivatives are simply set to 0.





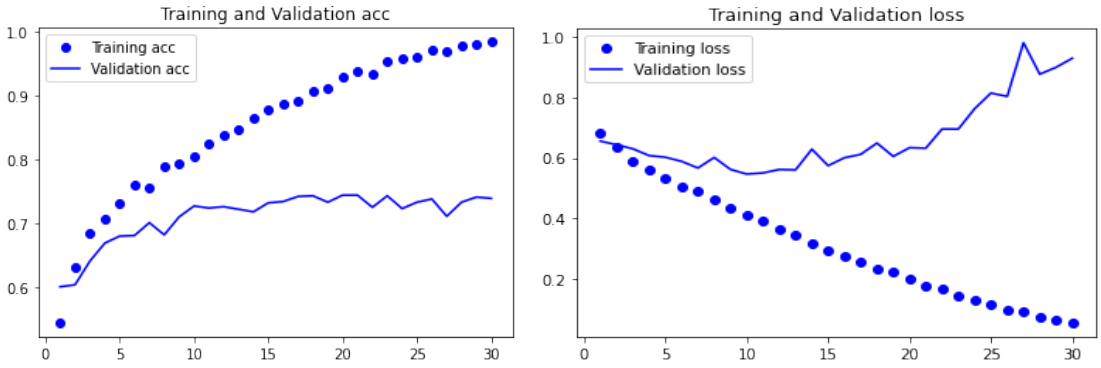
The last layer of the network is a single sigmoid unit, using binary cross-entropy as the loss function.

3.2 Data Preprocessing

Data must be converted to floating-point tensors before being fed into the neural network. Keras has a module for image processing: `keras.preprocessing.image`. It contains the `ImageDataGenerator` class to quickly create Python generators that process graphics files into batches of tensors. The output of the generator is a batch of 150-150 RGB images and binary labels with shape (20,). Each batch contains 20 samples (the size of the batch). The generator will generate these batches continuously, constantly looping through the images in the destination folder. The keras model uses the `fit_generator` method to fit the effect of the generator. The model has a parameter `steps_per_epoch` parameter: after extracting `steps_per_epoch` batches from the generator, the fitting enters the next round.

3.3 Model Fitting

training and validating network by original pictures 30 epochs. The validation accuracy is 75%, and the training accuracy is close to 100%. The reason for this is the module was overfitting these data. We will tune the model by improving the model architecture.



3.4 Data augmentation

Data augmentation would generate more training data from existing training samples, augmenting the data samples with various random changes that produce believable images. The model does not look at two identical images during training[6]

```
datagen = ImageDataGenerator(
    rotation_range=40,  # angle from 0-180
    width_shift_range=0.2,  # width range
    height_shift_range=0.2,
    shear_range=0.2,  # randomly change angle
    zoom_range=0.2,  # zoom range
    horizontal_flip=True,  # randomly flip half of the
pictures
    fill_mode="nearest"  # fill with nearest pixel
)
```

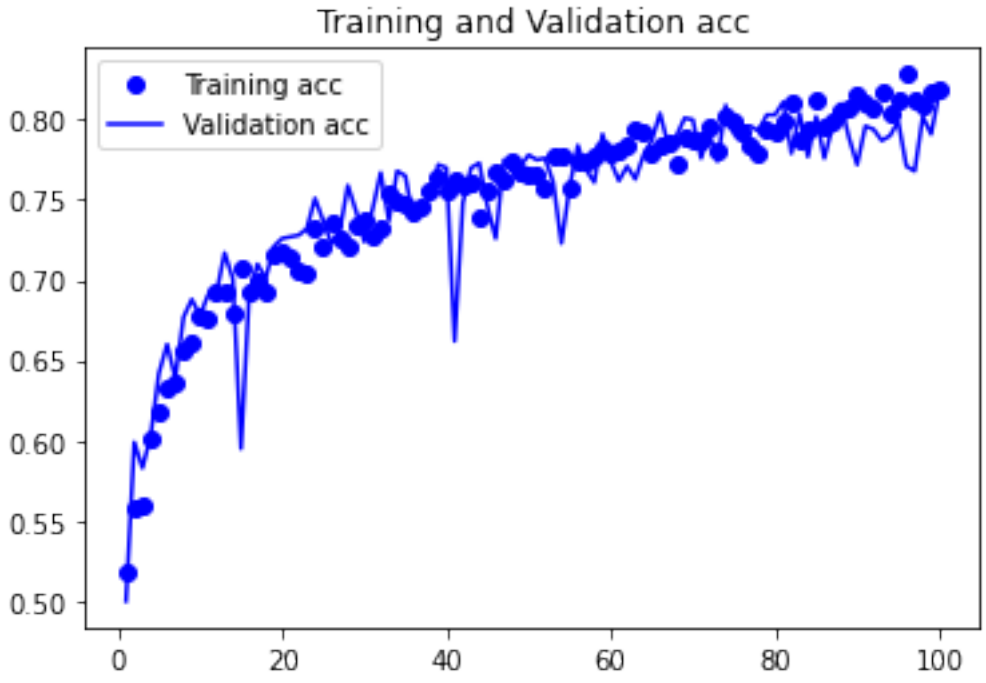
3.5 Dropout

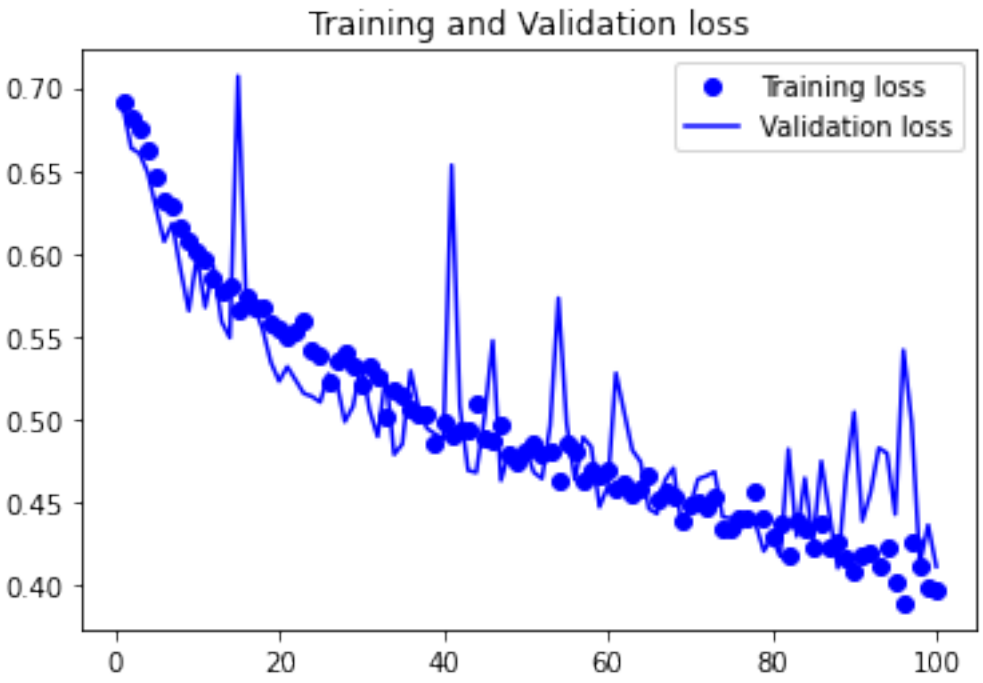
Data augmentation is used to train the network, but the input is still highly correlated, and overfitting cannot be eliminated. Consider adding a Dropout layer before the dense classification connector.[7] setting to zero the output of each hidden neuron with probability 0.5. The neurons which are “dropped out” in this way do not contribute to the forward pass and do not participate in backpropagation.

So, every time an input is presented, the neural network samples a different architecture, but all these architectures share weights. This technique reduces complex co-adaptations of neurons since a neuron cannot rely on the presence of other neurons. It is, therefore, forced to learn more robust features that are useful in conjunction with many different random subsets of the other neurons.

4 Result

We built a ConvNet using Keras framework to detect cats and dogs. Firstly, the network is trained and validated by original pictures with 30 epochs. We get 75% in validation accuracy; the training accuracy is close to 100%. To prevent the overfitting of these data. We modify train pictures by using data augmentation which changed these pictures randomly to make sure that our module will not receive duplicated pictures. And the validation accuracy after data augmentation is coming up to 80% and training accuracy is also closing to 80% after 100 epochs.




Training and Validation loss
Training loss
Validation loss
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0
20
40
60
80
100

Reference

[1]Karbhari, V. M., & Ansari, F. (2009). Synthetic aperture radar and remote sensing technologies for structural health monitoring of civil infrastructure systems. In *Structural Health Monitoring of Civil Infrastructure Systems*. essay, Woodhead Publishing Limited/CRC Press.

[2] *Convolutional Neural Networks (CNN) with Deep Learning*. (n.d.). HappiestMinds. Retrieved December 16, 2022, from https://www.happiestminds.com/insights/convolutional-neural-networks-cnns/

[3] *What is image classification?—ArcMap | Documentation*. (n.d.). What Is Image Classification?&Mdash;ArcMap | Documentation. Retrieved December 10, 2022, from https://desktop.arcgis.com/en/arcmap/latest/extensions/spatial-analyst/image-classification/what-is-image-classification-.htm

[4] Mandal, M. (2021, May 1). *CNN for Deep Learning | Convolutional Neural Networks*. Analytics Vidhya. Retrieved December 17, 2022, from https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/

[5] *Dogs vs. Cats | Kaggle*. (n.d.). Dogs Vs. Cats | Kaggle. Retrieved December 10, 2022, from https://www.kaggle.com/competitions/dogs-vs-cats/overview

[6] Team, P. (2021, September 9). *Image Classification with Convolutional Network - Algoritma Data Science School*. Algoritma Data Science School. Retrieved December 10, 2022, from https://algoritmaonline.com/image-classification-cnn/

[7] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017, May 24). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90. https://doi.org/10.1145/3065386