Convolutional Neural Networks — Image Classification

Zoili Eduardo Tapia Mejía
Faculty of electrical and electronic engineering
Universidad Veracruzana
Veracruz, México
zoyedu@gmail.com

***Abstract*—This document is about the convolutional neural network and how it was applied in a final Artificial Intelligence project. It is shown that a convolutional neural network can be used to identified between images because are a specialized type of artificial networks that use a mathematical operation specifically designed to process pixel data.**

***Index Terms*—Artificial intelligence, Image Recognition, CNN**

I. Introduction

A convolution extracts tiles from the input attribute map and applies filters to them to calculate new attributes, resulting in an output attribute map or convoluted attribute (which may have a different size and depth than the input attribute map). Convolutions are defined by two parameters:

- Size of tiles to be extracted (usually 3x 3 or 5x5 pixels).
- The depth of the output attribute map, which corresponds to the number of filters that are applied.

During a convolution, the filters (arrays of the same size as the tiles) slide effectively over the map grid of input attributes horizontally and vertically, one pixel at a time, and extract each corresponding tile (Fig.1)

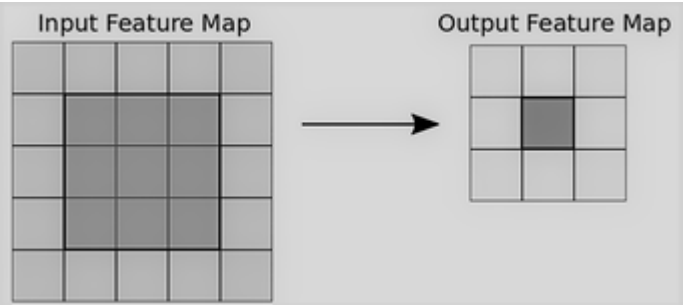

Fig. 1. A convolution 3x3 depth 1 performed on a map of 5x5 input attributes.

Convolutional Neural Network (CNN) is a class of artificial Neural Network most commonly applied to analyze visual imagery. They have applications in video and image recognition, image classification, image segmentation, medical analysis, language processing and more. CNN are simplified versions of multi-layer perceptrons that are fully connected networks, each neuron in one layer is connected to all neurons in the next layer. But back to the subject CNN are built-in convolutional layer that reduces the high dimensionality of images without losing its information.

Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the visual cortex. CNNs use little preprocessing compared to other image classification algorithms meaning that the network learns to optimize the filters through automated learning, where traditional algorithms are hand-engineered filter. This independence from prior knowledge and human intervention in feature extraction is a major advantage.

In a CNN, the input is a tensor with a shape: (number of inputs) × (input height) × (input width) × (input channels). After passing through a convolutional layer, the image becomes abstracted to a feature map, also called an activation map, with shape: (number of inputs) × (feature map height) × (feature map width) × (feature map channels).

II. Literature review

A. Background

To talk about the CNN, we must first talk about the history of deep learning that dates back to 1943, when Walter Pitts and Warren McCulloch created a computer model based on the neural networks of the human brain. They used a combination of algorithms and mathematics they called “threshold logic” to mimic the thought process.

The first ”Convolutional neural networks” were used by Kunihiko Fukushima who designed a neural networks with multiple pooling and convolutional layers, he developed an artificial neural network, called Neocognitron (Fig 2.), which used a hierarchical, multilayered design. This design allowed the computer the “learn” to recognize visual patterns.

While a neural network could be slow compared to a support vector machine, neural networks offered better results using the same data. Neural networks also have the advantage of continuing to improve as more training data is added. By 2011, the speed of GPUs had increased significantly, making it possible to train convolutional neural networks “without” the layer-by-layer pre-training. With the increased computing speed, it became obvious deep learning had significant advantages in terms of efficiency and speed.

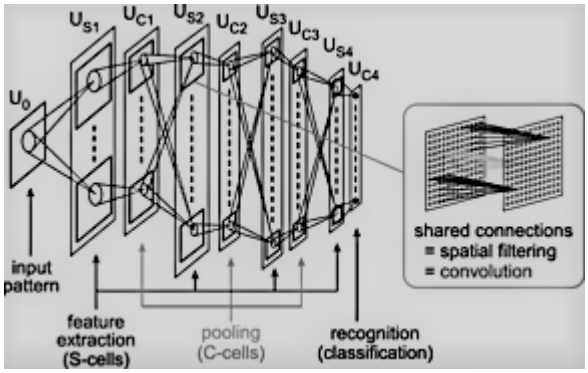

Fig. 2. The architecture of the neocognitron.

B. Related investigations

- "An Analysis Of Convolutional Neural Networks For Image Classification" This paper presents an empirical analysis of theperformance of popular convolutional neural networks (CNNs) for identifying objects in real time video feeds.[1]
- "Bag of Tricks for Image Classification with Convolutional Neural Networks" examine a collection of training procedure and model architecture refinements that improve model accuracy but barely change computational complexity.[2]
- "Advancements in Image Classification using Convolutional Neural Network" explanation of different CNN architectures for image classification. Through this document, we have shown progress on CNN from LeNet-5 to the latest SENet model. We discussed the model description and training details for each model. We also compared those models.[3]
- "Deep convolution neural network for image recognition" Deep learning can be parameterised for speed convergence with a minimal error rate, Methodology based on efficient convolution neural network (CNN) with five phases, Automatic extraction of image features for CNN-based classification and diagnosis.[4]
- "Analysis of Convolutional Neural Network based Image Classification techniques" Deep learning-based image classification system suggests DenseNet based model to classify the images effectively [5]

C. Architecture

CNN architectures consist of a few distinct layers. In all cases, the layers take as input a 3D volume, transform this volume through differential equations and produce a 3D volume, some layers require hyperparameter settings.

1) Input Layer: The first layer of each CNN used is 'input layer' which takes images, resize them for passing onto further layers for feature extraction

- The raw pixel values of an image represented as a 3D matrix
- Dimensions W x H x D, where depth corresponds to the number of color channels in the image.

2) Convolutional Layer: The next few layers are 'Convolution layers' which act as filters for images, hence finding out features from images and also used for calculating the match feature points during testing.

- Convolutional Layers will compute the output of nodes that are connected to local regions of the input matrix.
- Dot products are calculated between a set of weights (commonly called a filter) and the values associated with a local region of the input.

3) ReLu (Activation) Layer:

- The output volume of the Convolutional layer is fed to an elementwise activation function, commonly a Rectified-Linear Unit (ReLu).
- The ReLu layer will determine whether an input node will 'fire' given the input data.
- This 'firing' signals whether the convolution layer's filters have detected a visual feature.
- A ReLu function will apply max(0,x) a function, thresholding at 0.
- The dimensions of the volume are left unchanged.

4) Pooling Layer: The extracted feature sets are then passed to 'pooling layer'. This layer takes large images and shrink them down while preserving the most important information in them. It keeps the maximum value from each window, it preserves the best fits of each feature within the window

- A down-sampling strategy is applied to reduce the width and height of the output volume.

5) Fully-Connected Layer: The final layer is the fully connected layers which takes the high-level filtered images and translate them into categories with labels.

- The output volume, i.e. 'convolved features,' are passed to a Fully-Connected Layer of nodes.
- Like conventional neural-networks, every node in this layer is connected to every node in the volume of features being fed-forward.
- The class probabilities are computed and are outputted in a 3D array with dimensions: [1x1xk], where k is the number of classes.

III. Design and implementation

For the implemented model we worked on the application of google "Colab" that allows any user to write and execute arbitrary Python code in the browser and that offers free access to computer resources, such as GPUs. Is a free Jupyter Notebook environment that requires no configuration and runs completely in the cloud without the need to install or need something.

The main libraries (Fig. 3) used were Tensorflow, keras, numpy. Allowing machine learning and artificial intelligence with a variety of tasks, but has a particular focus on training and inference of deep neural networks and with keras experimentation in more or less short time with deep learning networks.

First we started collecting several images for the training and testing of the artificial intelligence 10,000 images in total,

```
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from IPython.display import Image
from keras.preprocessing.image import ImageDataGenerator
import numpy as np
from keras_preprocessing import image
```

Fig. 3. Libraries used in the proyect

when creating this database of images will serve us for the analysis of fish, Clownfish and Angel fish were chosen for this project, after spending time collecting and downloading images I pass them directly to my drive so it can be used in colab, for that it is imported using the code(Fig. 4).

```
from google.colab import drive
from IPython.display import Image
drive.mount('/content/drive')
```

Fig. 4. Code to import Drive

For the creation of the convulational neural network, the first layers were created and the model was created using Sequential() allowing us to create a container to which we can later add other elements that will define the characteristics of our model.

For the creation of the first layer(Fig. 5) is added to the model Conv2d() This layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs. When using this class as the first layer in a model, it provides the keyword argument inputshape.

We replicate the code for the realization of the second layer of the neural network, being a simple image classification project, the use of the second layer gives us the option of more precision of the classification

And using Maxpooling2d helps us downsampling the input along its spatial dimensions (height and width) by taking the maximum value over an input window (of size defined by pool size) for each channel of the input. The window is shifted by strides along each dimension.

```
model = Sequential()
#Primer capa
model.add(Conv2D(32, (3, 3), input_shape = (64, 64, 3), activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
#Segunda capa
model.add(Conv2D(32, (3, 3), activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
```

Fig. 5. Creation of CNN

For the next part I applied flatten() that returns a copy of the collapsed array to a single dimension and dense implements activation is the element wise activation function passed as the activation argument, kernel is a weights matrix created by the layer, and bias is a bias vector created by the layer.(Fig. 6)

To train the neural network, I started with the importation of ImageDataGenerator, this allows us to generate such blocks, in addition to performing the technique called data augmentation, when a relatively small number of images is available, we can increase the number by modifying the original images.

```
model.add(Flatten())
model.add(Dense(128, activation = 'relu'))
model.add(Dense(1, activation = 'sigmoid'))
```

Fig. 6. Creation of CNN

It lets augment our images in real-time while our model is still training. I can apply any random transformations on each training image as it is passed to the model. This will make our model robust but will also save up on the overhead memory.

```
from keras.preprocessing.image import ImageDataGenerator

train_data = ImageDataGenerator(rescale = 1./255, shear_range =
                                0.2, zoom_range = 0.2, horizontal_flip = True)

test_data = ImageDataGenerator(rescale = 1./255)
```

Fig. 7. Training the Artificial Intelligence

Now you enter our database that is located on my drive, assigning the images a value of 64 pixels in width and height and Size of the batches of data and the class mode: Set to binary is for 1-D binary labels. 10,000 images were detected in the drive, 8,000 are training 2 classes and 2,000 are test corresponding to 2 classes.

```
training_set = train_data.flow_from_directory('/content/drive/MyDrive/Topicospescado/Basedatos/entrenamiento',
                                              target_size = (64, 64), batch_size = 32, class_mode = 'binary')

test_set = test_data.flow_from_directory('/content/drive/MyDrive/Topicos pescado/Basedatos/Testeo',
                                         target_size = (64, 64), batch_size = 32, class_mode = 'binary')

Found 8000 images belonging to 2 classes.
Found 2000 images belonging to 2 classes.
```

Fig. 8. Training the Artificial Intelligence

When we put intelligence to train, we put the function of compile(), using the optimizer Adam that is a string for an optimizer instance is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments. Loss function is used to find error or deviation in the learning process. Keras requires loss function during model compilation process. Accuracy is a metric creates two local variables, total and count that are used to compute the frequency with which y pred matches y true. This frequency is ultimately returned as binary accuracy: an idempotent operation that simply divides total by count.

Fit is the method used for the model training on the data set for the specified number of fixed epochs or iterations mentioned, already here is where the program trains and for this, epoch 30 were created to get you to train , the reason why the amount of data is limited for the analysis was that colab no longer allow so many immagenes to be analyzed in the same situation.

```
model.compile(optimizer = 'adam', loss = 'binary_crossentropy',
              metrics = ['accuracy'])
model.fit(training_set, steps_per_epoch = int(8000/40),
          epochs = 30, validation_data = test_set,
          validation_steps = int(2000/40))
```

Fig. 9. Training the Artificial Intelligence

IV. Results

For the end of the code, you select the photo you want to compare and search, after selecting, the model is analyzed to generate an array to decide between two fishes.

```
test_image = image.load_img('/content/drive/MyDrive/T
                            target_size = (64, 64))
test_image = image.img_to_array(test_image)
test_image = np.expand_dims(test_image, axis = 0)
result = model.predict(test_image)
training_set.class_indices
if result[0][0] == 1:
    prediction = 'Pez Payaso'
else:
    prediction = 'Pez Angel'

1/1 [==============================] - 0s 22ms/step
```

Fig. 10. Choosing and analyzing photo

Fig. 11. Angel Fish

Fig. 12. Clown fish

Showing the difference between clown fish and angel fish.

V. Conclusions and future works

For the creation of the project and the achievement of creating it, I guided myself in several tutorials on the subject, since we covered talking about neural networks during the semester but we did not get to talk about convulational networks. But finishing this project and I had to make the video and document explaining, is when I understood the whole issue of neural network.

I applied discovered certain errors that may have been implemented in the code. This project can be ampled to more animals, fish, parts, objects, diseases, applicable to other areas, helping patients with any disease.

References

[1] Neha Sharma, Vibhor Jain, Anju Mishra, An Analysis Of Convolutional Neural Networks For Image Classification, Procedia Computer Science, Volume 132, 2018, Pages 377-384, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2018.05.198.

[2] He, T. (2018, 4 diciembre). Bag of Tricks for Image Classification with Convolutional Neural Networks. arXiv.org. https://arxiv.org/abs/1812.01187

[3] F. Sultana, A. Sufian and P. Dutta, "Advancements in Image Classification using Convolutional Neural Network," 2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN), 2018, pp. 122-129, doi: 10.1109/ICRCICN.2018.8718718.

[4] Boukaye B., Bernard K., Fana T., Deep convolution neural network for image recognition, Ecological Informatics, Volume 48, 2018, Pages 257-268, ISSN 1574-9541, https://doi.org/10.1016/j.ecoinf.2018.10.002. (https://www.sciencedirect.com/science/article/pii/S1574954118302140)

[5] Aktaş, Y. Ç. (2022, 31 enero). Image Classification with Convolutional Neural Networks. Medium. https://towardsdatascience.com/image-classification-with-convolutional-neural-networks-12a7b4fb4c91

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] Lang, N. (2022, 25 octubre). Using Convolutional Neural Network for Image Classification. Medium. https://towardsdatascience.com/using-convolutional-neural-network-for-image-classification-5997bfd0ede4