MACHINE LEARNING III

CNN

Group Assignment I

Professor: Rubén Zazo Candil

Group: E

Date: 5/4/2021

Students:

1. Jaime Bardisa
2. Nuno Cardoso
3. Alvaro Céliz
4. Mariana Coca
5. Raquel Suñer
6. Eduardo Troncoso
7. Jules Vanthournout

1. CATS & DOGS

TODO 1. *Add a convolutional layer with the specifications defined in the code.*

TODO 2. *Flatten the output of the last convolutional layer to add a dense layer.*

TODO 3. *Add the dense layer (the one before the last output layer) with the*

specifications defined in the code.

The goal of the first part is to build a neural net to classify cats and dogs images. The first part that we did was an exploration of the example data with a total of 2,000 training images and 1,000 for validation.

Then, we started to build a small convnet from scratch based on the original image (input) and adding convolution nets of different filters and max-pooling layers with a 2x2 window.

Once we add the filters, we can flat the previous output using the function *Layers.Flatten()* and then, we created a fully connected layer (with 512 hidden layers and a ReLU activation). Finally, we created the model with the original image as an input and a layer with a single node and a sigmoid activation as an output.

The next step was data preprocessing with a normalization of the images in a 0-1 range and then, we train the images for 15 epochs. After that, we visualize intermediate representations to see if the convnet has learned well and at the end, we evaluate the accuracy and loss of the model in training and validation.

2. REGULARIZATION

TODO 1

Data Augmentation. Explain how it is working ImageDataGenerator. Specifically, explain what all the parameters, already set, and their respective values mean.One by one, from rotation_range to fill_mode.

First, we started with the concept of data augmentation to make the most of the training examples by augmenting the number of random transformations. By doing that, we reduce the overfitting of the model.

The function that helps with data augmentation is *ImageDataGenerator* which generate batches of tensor image data with real-time data augmentation; meaning that it allows you to randomly rotate images with different degrees to display a "new" image with the purpose of avoid overfitting. The different parameters are:

- *rotation_range:* A value represented in degrees from 0 to 180 to randomly rotate pictures.

- *width_shift* and *height_shift:* Ranges represented as a fraction of total width or height, which allow translating pictures horizontally or vertically.
- *shear_range:* Used to randomly apply shearing transformations.
- *zoom_range:* Used to randomly zoom inside pictures.
- *horizontal_flip:* Allows flipping half of the images horizontally, which is most relevant when horizontal asymmetry is not assumed (e.g. real-world pictures).
- *fill_mode:* used for filling in newly created pixels, which may appear after a width/height shift or rotation.

After applying the function, we made an example with 5 different pictures of the same cat picture and we added the data augmentation to the preprocessing step.

TODO 2

Dropout.

Another strategy to avoid overfitting is dropout. **Dropout** is an additional form of regularization, very useful for NN. It works by randomly "dropping out" units in a network for a single gradient step (there is a connection to ensemble models). The more drop out, the stronger the regularization (intermediate values are more useful while a drop out of 1 means that the model does not learn anything; it is useless). It is one of the key advances that has enabled several strong results that we've gotten and that has recently pushed deep learning to the forefront. The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting. Inputs not set to 0 are scaled up by 1/(1 - rate) such that the sum over all inputs is unchanged.
Finally, we retrain the model with the data augmentation and dropout in place and we evaluate the results with the accuracy and loss function.

TODO 3

Fit the final model following the specifications in the code.

Done in the collab file following the specifications.

3. STAND OF THE SHOULDERS OF GIANTS

TODO 1

Do some research and explain the inception V3 topology. Which database do they use for training it? How was trained?

To increase the accuracy of the model, we can use another 2 techniques: feature extraction and fine tuning. For the first one, we will use the Inception V3 model.

Inception V3 is a commonly used image recognition model, which uses the **ImageNet** database for training and can attain higher than 78.1% accuracy when training a CNN. It is composed of two parts: Feature extraction with a CNN and Classification composed by fully

connected layers. It extracts general features from input images in the first part, and simultaneously classifies them based on those features in the second part. This model can achieve several improvements by using Label Smoothing techniques, factorized 7 x 7 convolutions and an auxiliary classifier to propagate label information lower down the network. It can classify images into 1000 object categories on the pre-trained network, and as a result, it will have learned rich feature representations for a broad range of images.

TODO 2

Inception V3 is set by default to admit input images of (299, 299, 3) dimensions; but we want (and we will use) inputs of (150, 150, 3). If the net is already trained, how is that even possible?

When applying Inception V3, it is essential to commit to an image of 3 channel, as the ImageNet database contains 3 channels, in order to make changes in input sizes. To change the input images of (299, 299, 3) to (150, 150, 3), we need to compute *include_top = False* within the variable *input_shape*. This variable refers to the optional shape tuple and height and width should be higher than 75 for an optimal output.

The main reason why a new input size after the net has been trained is possible, is because convolutions created within Inception V3 do not take into account image-sizes. However, it is important to consider that if input image dimensions are too small the model could be at risk, as it could lead to a negative impact on loss/accuracy or to even reducing the tensor dimensions into nothing. For this reason, it is best to change the input dimensions in steps to avoid altering the alignment of the feature maps through the CNN.

Source: *https://stackoverflow.com/questions/66289419/can-inception-v3-work-with-image-size-150x150x3*

TODO 3

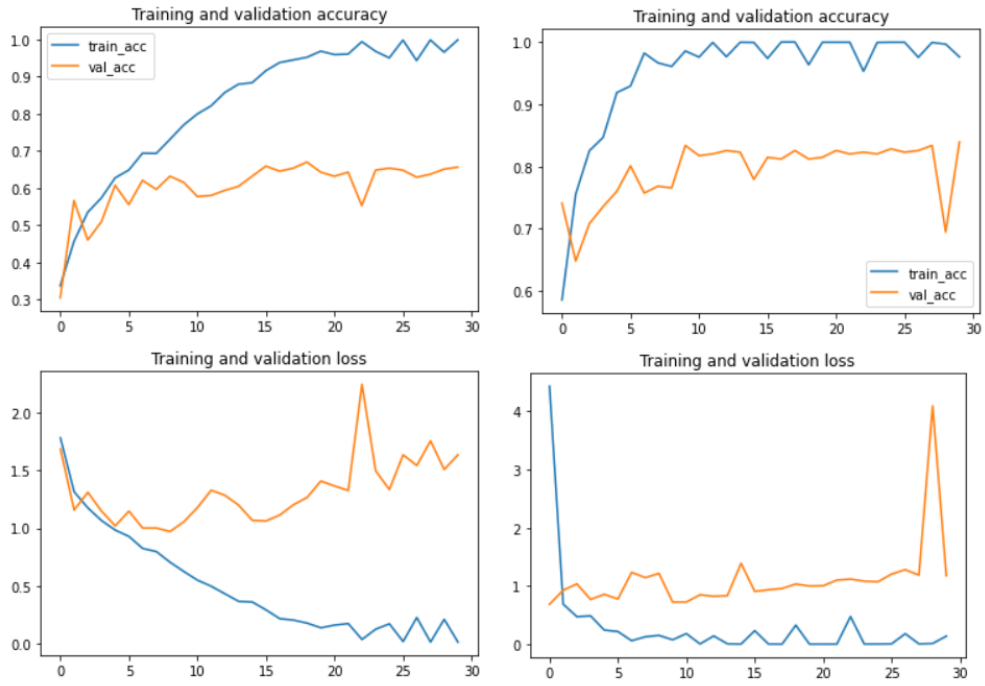
Let's change the database. Will Inception work well with different objects?

Yes, as long as these objects belong to the ImageNet dataset Inception V3 can be trained with them and allow us to develop a better model.

First, we apply the same steps as before to create a Neural Network for the flower's dataset, but with some changes in the template. We change the Input sizes to (100, 100, 3) and using the SoftMax activation since we will add a SoftMax layer for classification. Originally, we were using sigmoid but since as we are working with 5 classes instead of doing binary classification, we need to use SoftMax.

As well, for compiling the model we are using sparse categorical cross entropy instead of binary cross entropy since the task consists in multioutput classification instead of binary classification.

Finally, we evaluate again the model with the accuracy and loss function of the training and validation dataset. We can see a considerable improvement in validation accuracy and validation loss after applying inception V3.



4. Object Detection. YOLO Nets. [OPTIONAL]

TODO 1

YOLO is a state of the art, real-time object detection system. Its Mean Average Precision is on par with Focal loss but about four times faster. It also allows you to tradeoff between speed and accuracy by changing the size of the model.

Instead of applying the model to an image at multiple locations and scales, to then, consider as detections the high scoring regions (which is the typical approach by prior detection systems which repurpose classifiers to perform the detection), YOLO applies a single neural network to the image. It divides the image into regions, to then predict bounding boxes and probabilities for each. The bounding boxes receive a weight from the predicted probabilities.

Main advantages over classifier-based systems:

- Looks at the whole image at test time so the predictions are benefited from the global context in the image.

- Makes predictions with a single network evaluation while other systems such as R-CNN require thousands for one image. This allows YOLO to be way faster than other systems, over a thousand times faster than R-CNN and one hundred times faster than FAST R-CNN.

TODO 2

In order to demonstrate the YOLO V5, we have selected the Aquarium dataset (obtained in Roboflow) which contains images about different types of fishes. Thanks to the neural net, we can detect the different types of images with high precision and accuracy. In the notebook, we have explained the different steps of how to create the neural net and then, we did the training part (it takes around 30 minutes) and finally, we used different metrics to evaluate the model.