Jimmy Harvin & Charles Wallis
CS4375 - Professor Karen Mazidi
Dec 4, 2022

Image Classification

Sequential Models

Sequential models use serial data, where objects in a data set have a certain order. Examples could be DNA sequences, temperature sequence, or audio over time data. The sequential model that is provided by pytorch adds layers sequentially, and this architecture can be modified to achieve either of the following two architectures as well. These models will take in sequences of data, then output sequences of data. For dog breed identification this may not seem like a good idea intuitively, but since computers can read images as a sequence of numbers by pixels, this can be applied to classifying dog breeds.

RNN

Recurrent neural networks are a cyclic architecture that deals with dynamic sequences of data. RNN signals can be one-dimensional, two-dimensional, and three-dimensional depending on the domain where domain is defined by ‘where it is mapped from’ and ‘where it is mapped to’ and since a domain is just a temporary input to X, dealing with ordered data is basically the same as dealing with one-dimensional data. RNN provides an internal memory for the system, and it can help with predicting patterns and subsequent values such as in voice or text recognition. LSTM is a type of RNN model that can provide a solution to the vanishing gradient problem, which is a problem that occurs during the process of updating the weight of networks, where the gradient continues to vanish over time and the state of past not having any effect on the current training.

CNN

In convolutional neural networks, feature maps are extracted from a number of convolutional layers and dimensions are reduced through sampling to only gather important information about the feature map. This is an essential technology used in most computer vision fields, such as image classification. It requires a low computation amount and shows great performance compared to existing mlps. ResNet is often used, because it performs very well when transfer learning is applied to pre-trained weights. In our notebook, ResNet50 is used. It adds residual value for each block bounded by Convolution -> Batch Normalization -> ReLU. Training goes through hundreds of convolution layers yet it performs well because of the residuals. One problem that CNN models have is that it needs a large amount of input data to effectively train. That is why we chose to use ResNet, a pre-trained model, to perform transfer learning and use it to identify dogs.

Transfer Learning

Transfer learning is a technique for reusing a model that has already learned a particular task, to perform another one. These models tend to train faster than building a model from scratch, and often perform better too. This is possible because the network has been trained with a lot of features from many images, and the deeper the network becomes, it will learn more types of features. By utilizing transfer learning, we can bring ResNet directly and enter our data to create a classifier model. We can reset the top layer as we want, and it only trains using the data entered after it has been fully connected. After that, we can fine tune the final layer to increase accuracy.

Performance

The performances of the algorithm itself when we ran the code seemed not quite right, they were very far from what we expected. It definitely is possible that there has been a mistake in how we entered our data to the model. The CNN ResNet50 Transfer learning model performed greatly, but it required a lot of computations and time. Five epochs of training and validation took three hours to run (could have been a google colab limitation), and we were not able to allocate enough time to do another run with more epochs for a better result. We definitely saw improvements for different epoch counts though, and it is certain that this model can perform well after several more training epochs with some adjustments to the learning rate.

Formatted tables from data section of notebook

```
[ ] print(train.head(5).to_string())
    listOfTrain = list(train['id']) #list training data, without ".jpg"
```

	breed	id	affenpinscher	afghan_hound	african_hunting_dog	airedale	american_staffordshire_terrier
5657		8e802d5459b5e857e832c721d6cd3a35	0.0	0.0	0.0	0.0	0.0
6577		a57247bc5d572abd95f1aea215b7de77	0.0	0.0	0.0	0.0	0.0
7455		bb4aa91f61fa6e06dbc1a3869738b889	0.0	0.0	0.0	0.0	0.0
4552		71bd232f6f89bf4ea89af19e63a75378	0.0	0.0	0.0	0.0	0.0
9234		e7fda54b5e96c6048dad68414568f142	0.0	0.0	0.0	0.0	0.0

```
[ ] print(valid.head(5).to_string())
    listOfTest = list(valid['id']) #list of testing data, without ".jpg"
```

	breed	id	affenpinscher	afghan_hound	african_hunting_dog	airedale	american_staffordshire_terrier
5		002211c81b498ef88e1b40b9abf84e1d	0.0	0.0	0.0	0.0	0.0
14		0075dc49dab4024d12fafe67074d8a81	0.0	0.0	0.0	0.0	0.0
26		00a862390341c5be090dd72bd2bc19ef	0.0	0.0	0.0	0.0	0.0
28		00ba244566e36e0af3d979320fd3017f	0.0	0.0	0.0	0.0	0.0
37		0100f55e4f0fe28f2c0465d3fc4b9897	0.0	0.0	0.0	0.0	0.0