ANLY-590 Assignment 3

November 2020

1.Autoencoder

A convolutional autoencoder is a particular flavor of autoencoder where we use convolutional layers instead of Dense layers. We have previously applied autoencoders to images using only Dense layers and the result worked fairly well. However, the local spatial correlations of images imply that we should Be able to do better using convolutional layers instead of Dense layers.

Build and fit a convolutional autoencoder for the CIFAR10 dataset.
The components of this network will be many of the same pieces we've used with convolutional classification networks: Conv2D, MaxPooling, and so on. The encoder part of the network should run the input image through a few convolutional layers of your choice. The decoder part of the network will utilize UpSampling2D to get the representation back to the original image size.

An example to guide your thinking can be found toward the bottom of this Post **https://blog.keras.io/building-autoencoders-in-keras.html**.
DO NOT JUST COPY THIS CODE AND TURN IT IN. BE CREATIVE, COME UP WITH YOUR OWN VARIATION.

After training your network, visualize some examples of input images and their decoded reconstruction.

2. Image Classification

We'll continue to use the CIFAR10 dataset and build a deep convolutional network for classification.

2.1 Deep CNN

Build a deep CNN to classify the images. Provide a brief description of the architectural choices you've made: kernel sizes, strides, padding, network depth.Train your network end-to-end. Report on your model's performance on the training set and test set.

2.2 Transfer Learning

Repeat the same task, but this time utilize a pre-trained network for the majority of your model. You should only train the final Dense layer, all other weights should be fixed. You can use whichever pre-trained backbone you like (ResNet, VGG, etc). Report on your model's performance on the training set and test set.

3. Text Classification

While images contain local spatial correlations and structure, many other datasets contain temporal correlations. Examples include time series and discrete sequences such as text. In this problem, we will tackle the task of text classification in the context of natural language.

Background. In this problem, we will build models that read text segments (messages) and identify whether they are **SPAM** or **HAM**.

Wikipedia describes **SPAM** as "the use of electronic messaging systems to send unsolicited bulk messages, especially advertising, indiscriminately."

The term **'HAM'** was originally coined by SpamBayes sometime around 2001 and is currently defined and understood to be "E-mail that is generally desired and isn't considered SPAM."

Dataset. The dataset consists of $\sim$ 5500 messages along with binary labels(SPAM or HAM) and is already preprocessed. So basically each sample is like **[MESSAGE, LABEL]**.

3.1 RNN

Build and train a Recurrent Neural Network to solve this text classification task. You can use any type of RNN you wish (SimpleRNN, GRU, LSTM).

3.2 CNN

Build and train a 1D CNN for this text classification task. We recommend you do a character-level convolution (with character embeddings). You might gain some insight and inspiration from these text classification approaches:

1. **http://www.aclweb.org/anthology/D14-1181**
2. **https://arxiv.org/abs/1702.08568**

Tips: after splitting every character in each training sample, the maximum length of training samples can be really big. If you choose to only do the padding trick to all the samples, it might raise OOM issues. So instead of padding only, you can also cut each sample at a certain point, which can make training more efficient and feasible while using CNN.

3.3

Be sure to directly compare your two methods with an ROC curve or similar validation method. Don't forget to create a train-test split.