ANLY-590 Assignment 2

October 2018

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 Fashion MNIST 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`.

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

2 Image Classification

We'll continue to use the Fashion MNIST 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 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 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 cybersecurity.

Background. When malware infects a host computer, it often needs to reach out to an outside server for further instructions or to download additional payloads. This outside server is called a Command-and-Control server (C2). The malware needs to send a specific communication to the C2 server, thus the C2 server needs to have a registered IP address or associated web domain so that it can be reached. Therefore, being able to identify web domains that are likely related to malware C2 can be a valuable cyber defense.

Dataset. Fortunately, security researchers have already identified and logged a large number of malicious URLs. Additionally, we can catalog common "benign" URLs just from typical web behavior (these would include things like facebook.com and amazon.com). Hence, we have a labeled dataset for text classification which can be downloaded here:

- `https://s3.amazonaws.com/anly-590/url-classification/benign-urls.txt`
- `https://s3.amazonaws.com/anly-590/url-classification/malicious-urls.txt`

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. You might gain some insight and inspiration from these text classification approaches:

- `http://www.aclweb.org/anthology/D14-1181`
- `https://arxiv.org/abs/1702.08568`

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.