PROBLEM SET 4: APPLIED MATHEMATICS 216

Due: Friday March 4,2022 at 11:59pmt

Problems.

(1) **The Ising Model, continued** We will continue our discussion of the Ising Model. You will recall the Ising class we gave you in week 2, that allowed simulations of the 2d Ising Model on a square lattice. Now we also provide an Ising class for doing simulations on a triangular (or hexagonal) lattice. The number of nearest neighbors of a given node in a triangular lattice is 6 instead of 4. It is well known that the transition temperature for the phase transition changes between the different lattices, but the basic physics of the phase transition remain the same.

We now are providing 2 different datasets: the first contains spin configurations for the square lattice of the Ising models at four different temperatures; the second contains spin configurations for the triangular lattice at four temperatures. The temperatures for square lattice are $T = 1.5, 2.1, 2.4, 3.5$. $T = 2.5, 3.2, 3.8, 5$ for triangle lattice.

(a) Train a fully connected neural network to do the classification, on both datasets.

(b) Train a convolutional neural network to do the classification, on both datasets. Make a table of your performance numbers for (a) and (b). Try to optimize the performance of your models and compare the result.

(c) We have provided a test set of 10 spins configurations for each of the two problems. Each of the spin configurations is *not necessarily* at the temperatures of the training sets. Calculate your best estimate of the temperatures of these spin configuration. Upload your results to Kaggle. [Hint: A direct fingerprint of temperature is the distribution of spin up and down, because you can image that the spins fluctuate more violently at higher temperature. Can you take distribution into account when you build the model to estimate temperature? This may help you win the kaggle]

(d) **Transfer Learning**. As we emphasize in class, one can freeze the training of the bottom layers of a network and retrain the top part of the network to adopt to a new situation. Use your CNN that you trained on the squarelattice data to do transfer learning on the triangular lattice data. How does the performance compare to that of the direct methods? Add the performance numbers for transfer learning in your table from Part (a). Note that the training time and number of training examples needed for transfer learning is far less than that for the direct optimization. For example, is 50 triangle example sufficient for the re-training process? Use your transfer learning result to predict the transition temperature of triangle lattice Ising model, as demonstrated in this Nature Physics publication.
As a guideline, you may like to just change the last 'Dense' layer with 'softmax' activation when you do the transfer learning. Other choices are also OK.

(2) **Using Inception to Solve Scientific Problems** A series of deep models are made available through Keras Applications, along with pre-trained weights. These models can be used for prediction, feature extraction, and fine-tuning. Here we will add extra layers to the top of inception and retrain them on two different scientific problems:

(a) **The Ising Model** – try your best to show that the square lattice data can be trained perfectly with inception's 2048*1 embedding vector.

(b) **Rayleigh Bernard Convection**, in which a flow is heated from below and cooled from top, is one of the paradigmatic system in fluid dynamics. When the temperature difference between the two plates (in dimensionless form Rayleigh number Ra) is beyond certain threshold, hot fluid tends to go up and cold fluid tends to go down, forming convection cells. What we supply here are the temperature snapshots from four different Ra, i.e., $Ra = 10^{14}$ as class 0, $Ra = 10^{13}$ as class 1, $Ra = 10^{12}$ as class 2, and $Ra = 10^{11}$ as class 3. The flow you see is highly turbulent; not only there are big convection cells but also lots of small vortices. The original dataset is around 4000*2000. We have already downsampled the data into the zip file 'fluid-org'.

(b1) Train the data in 'fluid-org.zip' with inception. Show that these images can be classified into different Ra classes nicely with inception. Take the length 2048 embeddings from the Inception model first. Then visualizing how the embeddings distribute using a two component PCA or two component T-SNE, whichever you prefer. Then use any of the previously learned method to train a classifier using the embeddings as input. Note that T-SNE normally gives you better separation between data clusters.

(b2) For advanced use of trainsfer learning from the pre-trained models such as fine-tuning, we need to do the transfer learning in-place, by building a network consists of the Inception and your classifier layers. Freeze the part you take from Inception, train the model and report the accuracy. Then do the fine-tuning. Report how much increase of accuracy you can manage to get. Fine tuning by making the top few layer of the Inception model trainable instead of freezing all the layers. Due to the slowness of training, unleash the layers one by one. Make comments about how the accuracy change. It is highly recommended that you train this on Google Colab with the GPU activated.

(b3) Explore the potential of transfer learning on cropped data 'fluid-crop', which are randomly choosen regions of 100*100 pixels from each original 4000*2000 pictures, i.e.,just around 1% of the original picture!. You can use either method you use in (b1) or (b2).

(b4) Build your own classifier for (b2) and (b3) without using Inception. Compare the performance of your own classifier with the result in (b2) and (b3).

(b5) Continue from (b3), construct two examples, each of which uses a different layer's output as the embedding. From the over 300 layers in Inception, pick one at around the 100th layer and one at around 200th layer. The exact layer you pick is based on your preference. Show the following two things: (i) The distributions of the embeddings similar to what you've done in (b1). Together with the result you get in (b1), comment the similarity and difference between what you get using the three embedding layers. (ii) What is the test accuracy of the three classifiers?
For speeding up the training you can choose to get the embeddings first and put those into a classifier, as you did in (b1).