CS 449 Final Project Update

Due: Feburary 24, 2023 at 11:59pm

1. Names and Net IDs

Erick Mungai - mmm1176

Rodney Reichert - rdr3218

Perry Benyella - pbe2757

2. Abstract

Write a new abstract that describes the motivation for your project and proposed methods. This should reflect any changes you've made, but you'll explain those changes later.

Our final project seeks to use a variety of architectures to detect the presence of metastatic tissue from histopathological scans of lymph node sections. We will use both a CNN and a Residual Network(ResNet) to generate feature vectors for the images and then vary the output network between an RNN, LSTM network and a fully connected feedforward neural network. We will then compare the performance of these networks to see which performs best. The tentative combination of networks will be CNN + RNN, CNN + LSTM, CNN + feedforward neural network, ResNet + RNN, ResNet + LSTM, ResNet + feedforward neural network.

3. Big Changes

We have decided to keep our original proposal with no changes.

4a. Describe your dataset(s)

We will use the PatchCamelyon (PCam) dataset, which contains a total of 327,680 color images (96 x 96px) extracted from histopathologic scans of lymph node sections. PCam is derived from the Camelyon16 Challenge, which contains 400 H&E stained WSIs of sentinel lymph node sections. The slides were acquired and digitized at 2 different centers. PCam marries the clinically-relevant task of metastasis detection into a straight-forward binary image classification task, much like MNIST. The feature structure for this dataset can be expressed through the following dictionary:

```
FeaturesDict({
    'id': Text(shape=(), dtype=string),
    'image': Image(shape=(96, 96, 3), dtype=uint8),
    'label': ClassLabel(shape=(), dtype=int64, num_classes=2),
})
```

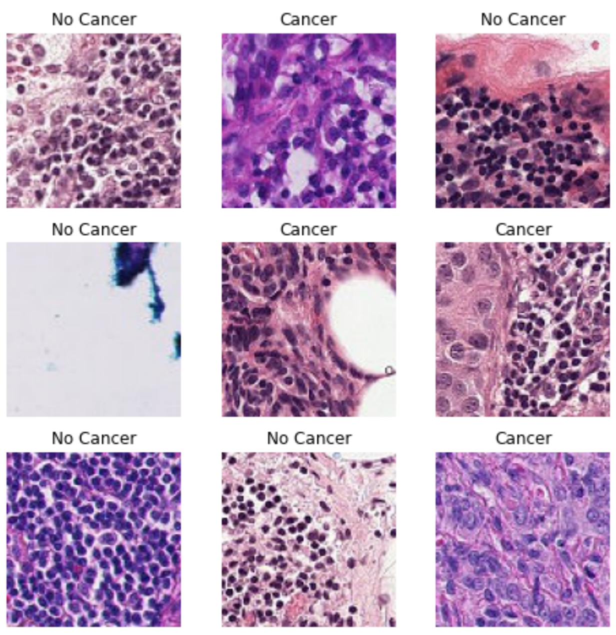
Each datapoint has a unique id, a matrix of uint8 values representing the RGB values for each of the 96^2 pixels of the image as well as a binary label.

The official website for the PatchCamelyon benchmark gives the following two references with regards to the dataset:

```
1. B. S. Veeling, J. Linmans, J. Winkens, T. Cohen, M. Welling. "Rotation
   Equivariant CNNs for Digital Pathology". arXiv:1806.03962
2. Ehteshami Bejnordi et al. Diagnostic Assessment of Deep Learning
   Algorithms for Detection of Lymph Node Metastases in Women With Breast
   Cancer. JAMA: The Journal of the American Medical Association, 318(22),
   2199–2210. doi:jama.2017.14585
```

```
## 4b. Show some data
"""
    *Demonstrate that you have made at least some progress with getting your
     dataset ready to use. Load at least a few examples and visualize them
     as best you can*
"""

%pip install h5py

import torch
from torch.utils.data import Dataset
from torchvision import datasets
from torchvision.transforms import ToTensor
import matplotlib.pyplot as plt
import matplotlib.image as mpimg


## loading training data eats up all avalable ram

# training_data = datasets.PCAM(
#     root="data",
#     split='train',
#     download=True,
#     transform=ToTensor()
# )

test_data = datasets.PCAM(
    root="data",
    split="test",
    download=True,
    transform=ToTensor()
)

labels = {
    1: "Cancer",
    0: "No Cancer"
}
figure = plt.figure(figsize=(8, 8))
cols, rows = 3, 3
for i in range(1, cols * rows + 1):
    sample_idx = torch.randint(len(test_data), size=(1,)).item()
    img, label = test_data[sample_idx]
    img = img.permute(1, 2, 0)
    figure.add_subplot(rows, cols, i)
    plt.title(labels[label])
    plt.axis("off")
    plt.imshow(img)
plt.show()
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: h5py in /usr/local/lib/python3.8/dist-packages (3.1.0)
Requirement already satisfied: numpy>=1.17.5 in /usr/local/lib/python3.8/dist-packages (from h5py) (1.2
    800875929/? [00:19<00:00, 40609863.25it/s]

    3040/? [00:00<00:00, 107396.79it/s]
```



5. Updated Methods

This is a binary image classification task. We will use two architectures, a CNN and a ResNet to generate feature vectors for the images. CNN are a type of neural network that is widely used for image analysis. It was popularized by AlexNet which won the 2012 ImageNet competition. Residual Networks were envisioned by Kaiming He from Microsoft Research in 2015. They use skip connections in their residual block which help in solving the vanishing gradient problem in deep CNNs. Since this is a standard binary classification task, we will use Binary Cross Entropy as our loss function. BCE performs best for these kinds of problems. We will use an appropriate train-validation-test split which will enable us to gauge the model's ability to generalize to new data.

6. Previous Deliverables

Copy the deliverables from your proposal and write a sentence saying whether you completed this goal, made progress on this goal, or abandoned this goal. Unless you completed the goal, give an explanation of how it went.

6.1 Previous Essential Goals

- We will design a CNN and use it to generate feature vectors for the x-ray scans.
- We will then vary the output network between an RNN, LSTM and a fully connected feedforward neural network and benchmark the performances of all of them.

6.2 Previous Desired Goals

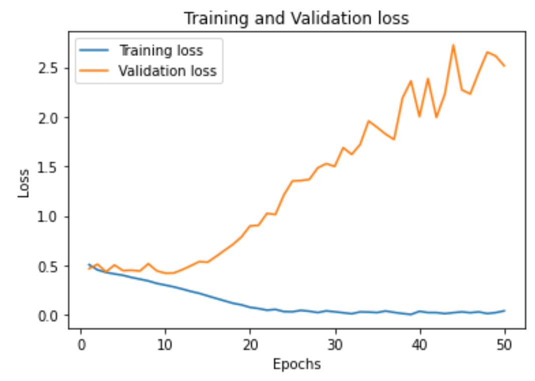
- Since a ResNet solves the vanishing gradient problem in deep CNNs, we will switch out the CNN for a ResNet(either design our own or use a pre-trained one) to generate the feature vectors.
- We will then vary the output network the same way we did with the CNN and benchmark the performance.

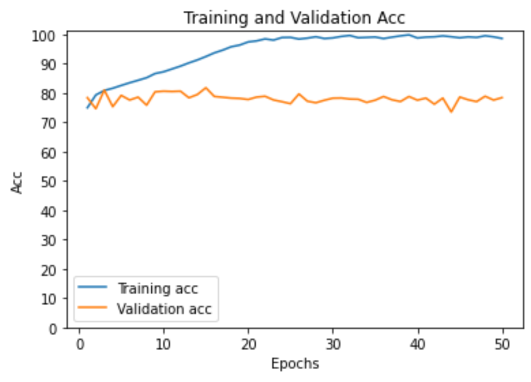
6.3 Previous Stretch Goals

- We will conduct a hyperparameter search for the best performing architecture combinations and see how changes in these parameters affect the performance, with the intention of improving performance even more.
- We will test our best performing architecture on a multi-class classification task to see whether it yields the same results.

```
# 7. Results So Far
"""
  Include some initial empirical results on how your model(s) perform on your
  dataset(s). Ideally, you should not write all your code here, rather you
  should import from other code in your GitHub repository and just show some
  initial results. If it's easier, you can just include plots and a description
  of how you made them, rather than runnable code.
"""
```

```
"""

import matplotlib.pyplot as plt
loss_train = [0.508279025554657, 0.4556257724761963, 0.43081802129745483, 0.4148216247558594, 0.40155887603759766, 0.37974247336387634, 0.
loss_val = [0.4680269956588745, 0.5125997066497803, 0.43674135208129883, 0.5049386620521545, 0.4489496946334839, 0.4522791802883148, 0.444
epochs = range(1,51)
plt.plot(epochs, loss_train,  label='Training loss')
plt.plot(epochs, loss_val, label='Validation loss')
plt.title('Training and Validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()


acc_train = list(map(lambda a: a * 100, [0.74955, 0.79325, 0.8085, 0.8161, 0.8257, 0.8349, 0.8433, 0.85205, 0.86625, 0.8723, 0.8819, 0.891
acc_val = list(map(lambda a: a * 100,[0.784, 0.7462, 0.8098, 0.7532, 0.7918, 0.776, 0.7858, 0.7584, 0.804, 0.8062, 0.8052, 0.8062, 0.7836,
plt.plot(epochs, acc_train,  label='Training acc')
plt.plot(epochs, acc_val, label='Validation acc')
plt.title('Training and Validation Acc')
plt.xlabel('Epochs')
plt.ylabel('Acc')
plt.yticks(range(0,110, 10))
plt.legend()
plt.show()
```





7. Results So Far

We have defined classes for the CNN, RNN, and LSTM. We have handled the preprocessing for the PCAM in preparation for training the RNN and the LSTM. Above we have printed the train and validation loss along with the accuracy for the traditional CNN trained on ~%8 of the training data. Our code for training is under CNN.ipynb within our repository.

8. New Deliverables

For any deliverables that you did NOT complete, copy them into these lists below. Then add at least one goal per level.

8.1 New Essential Goals

- We will design a CNN and use it to generate feature vectors for the x-ray scans.
- We will then vary the output network between an RNN, LSTM and a fully connected feedforward neural network and benchmark the performances of all of them.
- Achieve at least 85% accuracy with one of our models (For reference)

8.2 New Desired Goals

- We will conduct a hyperparameter search for the best performing architecture combinations and see how changes in these parameters affect the performance, with the intention of improving performance even more.
- We will test our architecture on a multi-class classification task to see whether it yields the same results.
- Achieve at least 90% accuracy with one of our models (For reference)

8.3 New Stretch Goals

- Since a ResNet solves the vanishing gradient problem in deep CNNs, we will switch out the CNN for a ResNet (either design our own or use a pre-trained one) to generate the feature vectors.
- We will then vary the output network the same way we did with the CNN and benchmark the performance.
- Achieve at least 95% accuracy with one of our models (For reference)

9. Hopes and Concerns

We are hopeful going forward given our success with the traditional CNN. We have achieved promising results with our preliminary training.

The concerns unfortunately outnumber the hopes. The primary source of our concerns is hardware. Firstly, we are getting some very unique errors when working with the GPU. We are able to resolve many of them by switching to the CPU but it is a somewhat laborious process. Additionally, the size of our dataset makes loading it all nearly impossible and so we will have to load portions of the dataset one at a time to train the models.

10. References

A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional Neural Networks,” Advances in Neural Information Processing Systems, 01-Jan-1970. [Online]. Available: https://papers.nips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” arXiv.org, Dec. 10, 2015.[Online]. Available: https://arxiv.org/abs/1512.03385.