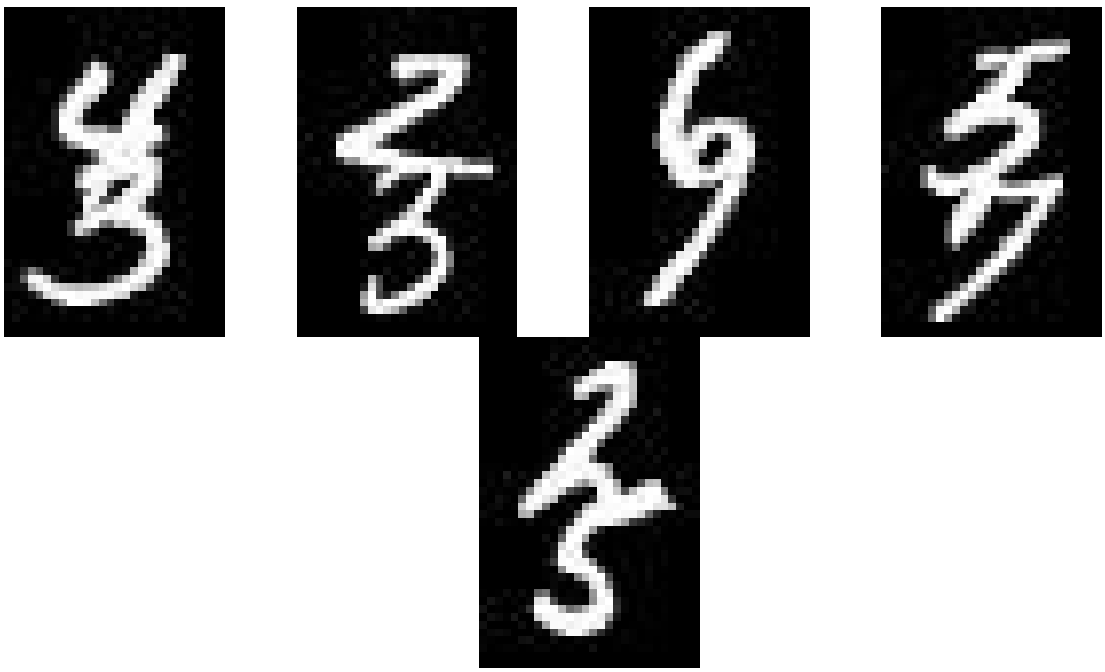
10. Overlapping, multi-digit MNIST

In this problem, we are going to go beyond the basic MNIST. We will train a few neural networks to solve the problem of hand-written digit recognition using a multi-digit version of MNIST.



You will be working in the files `part2-twodigit/mlp.py`, `part2-twodigit/conv.py`, and `part2-twodigit/train_utils.py` in this problem

---

In your project folder, look at the **part2-twodigit** subfolder. There you can find the

files **mlp.py** and **conv.py**. Your main task here is to complete the code inside the method `main` in these files.

Do the following steps:

- Look at `main` method in each file. Identify the training and test data and labels. How many images are inside the train and test data? What is the size of each image?
- Look at the definition of the MLP class in **mlp.py**. Try to make sense of what those lines are trying to achieve. What is `y_train[0]` and `y_train[1]`?
- Look at `train_utils.py`, particularly the `run_epoch` function.

Now given the intuition you have built with the above steps, complete the following tasks.

Fully connected network

0.0/5.0 puntos (calificable)

Complete the code **main** in **mlp.py** to build a fully-connected model with a single hidden layer with 64 units. For this, you need to make use of `Linear` layers in PyTorch; we provide you with an implementation of `Flatten`, which maps a higher dimensional tensor into an N x d one, where N is the number of samples in your batch and d is the length of the flattend dimension (if your tensor is N x h x w, the flattened dimension is $d = (h \cdot w)$). Hint: Note that your model must have two outputs (corresponding to the first and second digits) to be compatible with the data.

Available Functions: You have access to the `torch.nn` module as `nn`, to the `torch.nn.functional` as `F` and to the `Flatten` layer as `Flatten`; No need to import anything.

```
class MLP(nn.Module):

    def __init__(self, input_dimension):
        super(MLP, self).__init__()
        self.flatten = Flatten()
```

```
        # TODO initialize model layers here

    def forward(self, x):
        xf = self.flatten(x)

        # TODO use model layers to predict the two digits

        return out_first_digit, out_second_digit

```

Presione ESC y después TAB o haga clic afuera del editor de código para salir

Sin Responder

Enviar Ha realizado 0 de 25 intentos

Convolutional model

0.0/5.0 puntos (calificable)

Complete the code `main` in **conv.py** to build a convolutional model. For this, you need to make use of **Conv2D** layers and **MaxPool2d** layers (and perhaps Dropout) in PyTorch. Make sure that the last layer of the neural network is a fully connected (Linear) layer.

Available Functions: You have access to the `torch.nn` module as `nn`, to the `torch.nn.functional` as `F` and to the `Flatten` layer as `Flatten`; No need to import anything.

```
class CNN(nn.Module):

    def __init__(self, input_dimension):
        super(CNN, self).__init__()
        # TODO initialize model layers here

    def forward(self, x):

        # TODO use model layers to predict the two digits

        return out_first_digit, out_second_digit


```

Presione ESC y después TAB o haga clic afuera del editor de código para salir

Sin Responder

Enviar Ha realizado 0 de 25 intentos

Hyperparameter tuning

1 punto posible (calificable)

Next, change the parameters and settings of the models above. Train your model with various parameter settings. Some things you can try is to modify the learning algorithm from **SGD** to more sophisticated ones (such as **ADAM**) or you can modify the network architecture (number of layers or unit per layers, activation function, etc.). Something to ponder: What parameters or settings were more conductive to get a better model with greater generalization capability and lower error? Did extra training for a model always help or did the training accuracy plateau or even get worse after some point?

Finally, we will grade you on finding at least one architecture that achieves over 98% accuracy on both the validation and test set.

Please enter your **test accuracy** .

Enviar Ha realizado 0 de 5 intentos

Conclusion and What's Next

As you have seen in this project, neural networks can pretty successfully solve the MNIST task. In fact, since 2012, following the impressive performance of AlexNet on the ImageNet dataset, deep neural networks have been the standard in computer vision. As datasets went growing in size and complexity and as computing power became cheaper and more efficient, the trend has been to build deeper and bigger

neural nets.

The last part of the project has given you a hint as to why neural networks can be very versatile: by merely changing the output layer, you were able to train the network to predict overlapping MNIST digits. The same building blocks can be reused to build more complex architecture and solve more difficult problems. Using a deep learning framework like Pytorch makes this process even more accessible.

If you have access to a GPU, you can try implementing an object classification system with Resnet, which we have not covered in this course, and maybe expanding it to an object detection or an image segmentation system.

If you do not have access to a GPU, you can try renting resources from an online provider, such as Paperspace (<1$/hour) or Google Colab (free).

Discusión

Ocultar Discusión

Tema: Unit 3 Neural networks (2.5 weeks):Project 3: Digit recognition (Part 2) / 10. Overlapping, multi-digit MNIST

Add a Post

Mostrar todas las publicaciones por actividad reciente

?	Any hint on how to predict 2 digits instead of 1 digit When we want to predict one digit, we construct a final dense layer with output 10, but how a...	42
☑	Convolution meaning and activation? Hi. According to Erocha answer in this [post][1] I've execute this code: I=np.array([[0,0,0,0,0],[...	7
?	[staff] The grader is still processing Please help me check my situation...	2
?	Input to Layer1 programmatic calculation Can anyone help me understand how to programmatically calculate input to layer 1 instead o...	3
☑	[Staff]MLP, can't understad provided solution... Hi. Given solution class MLP(nn.Module): def __init__(self, input_dimension): super(MLP, self).__...	8

☑	Is it a valid solution. ReLU usage. Hi. For fully connected network I got following solution which was marked correct: class MLP(...	7
💬	Guidelines for Choosing Architecture & Hyperparameter Tuning? I was hoping that the course would provide some guidelines for choosing architecture (e.g. h...	12
💬	Need faster machine... Wah. I know what I want to do, but I'm still running one last training with a different model, a... Community TA	16
💬	What's the best accuracy you got? I was trying to optimize the model. I got 0.990171/0.987903 test error so far. It seems very ha... Community TA	35
💬	Thank You MIT -- this course is exactly what I was looking for. This course has been a fantastic challenge. After pushing through I've coded a softmax regres...	39
?	A summary of questions regarding digit recognition	7
💬	[STAFF] help with grader error I am getting this error. Any help appriciated There was a problem running your solution (Staff...	1
💬	[staff] issue with CNN class definition The CNN constructor is only passed the *product* of the number of image rows and columns... Community TA	4
💬	Convolutional model - need help I am not so clear what to do for this model and need some guidance. This post has helped m...	8
💬	The code complains data type not right on my local machine The error message: 75 out1, out2 = model(x) TypeError: argmax(): argument 'input' (position ...	1
💬	The status of job submission is stuck for "Fully connected network" Hi, The status of submission is stuck as "processing". Best	1
☑	Fully connected network - Not clear Any hints that once I have created a model in init, how should i use it for evaluation of 2 num...	6