Model used - pytorch convolution

jupyter mnist_kaggle_full Last Checkpoint: Yesterday at 11:19 AM (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [67]:
```
def conv2(ni,nf): return conv_layer(ni,nf,stride=2)
```

In [68]:
```
model = nn.Sequential(
        conv2(1,8),
        conv2(8,16),
        conv2(16,32),
        conv2(32,16),
        conv2(16,10),
        Flatten())
```

In [75]:
```
learn = Learner(data, model, loss_func = nn.CrossEntropyLoss(), metrics = accuracy)
```

In [76]:
```
learn.summary()
```

Out[76]:

Layer (type)	Output Shape	Param #	Trainable
Conv2d	[8, 14, 14]	72	True
ReLU	[8, 14, 14]	0	False
BatchNorm2d	[8, 14, 14]	16	True
Conv2d	[16, 7, 7]	1,152	True
ReLU	[16, 7, 7]	0	False
BatchNorm2d	[16, 7, 7]	32	True
Conv2d	[32, 4, 4]	4,608	True
ReLU	[32, 4, 4]	0	False
BatchNorm2d	[32, 4, 4]	64	True
Conv2d	[16, 2, 2]	4,608	True
ReLU	[16, 2, 2]	0	False
BatchNorm2d	[16, 2, 2]	32	True
Conv2d	[10, 1, 1]	1,440	True
ReLU	[10, 1, 1]	0	False
BatchNorm2d	[10, 1, 1]	20	True
Flatten	[10]	0	False

```
Total params: 12,044
Total trainable params: 12,044
Total non-trainable params: 0
```

A problem was the input data was a single channel data and could not work with already proven resnet models such as resnet34 or resnet50 since those models are designed for 3 channel input images.

Dataset:

a. Downloaded MNIST dataset provided by fastai URLs
    1. https://s3.amazonaws.com/fast-ai-imageclas/mnist_png
    2. Tried to use the mnist kaggle dataset which is train.csv and test.csv files and image pixel values are stored in csv file itself.
    3. Got into many hurdles trying to use this csv kaggle data.
        a. I could load this data in pytorch data structure using the panda library.
        b. But could not find a function in fastai which could take this array directly to be an image data and thus create an image databunch out of it.
    4. The images are single channel images. Total of 60000 training and 10000 test data (used for validation).
    5. This data is well arranged into labelled directories each inside training and test directories. Thus enabled to use ImageList.from_folder.
        a. Then split the data in 'training' and 'testing' using another function split_by_folder
        b. Then got labels by lable_from_folder.
        c. The label List thus obtained is used to create a databunch
        d. Normalize data

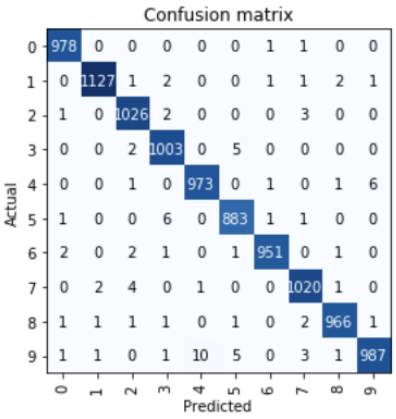
b. Categories/classes - 0,1,2,3,4,5,6,7,8,9
c. multi-class classification problem, i.e. logistic regression for multiple output classes.
d. Cost - $-\sum Yi\ log(Pi)\ +\ (1-Yi)log(1-Pi)$- loss_func = nn.CrossEntropyLoss()
e. Adam optimizer with momentum and all used internally by fastai library (hopefully).

Reference - https://github.com/fastai/course-v3/blob/master/nbs/dl1/lesson7-resnet-mnist.ipynb

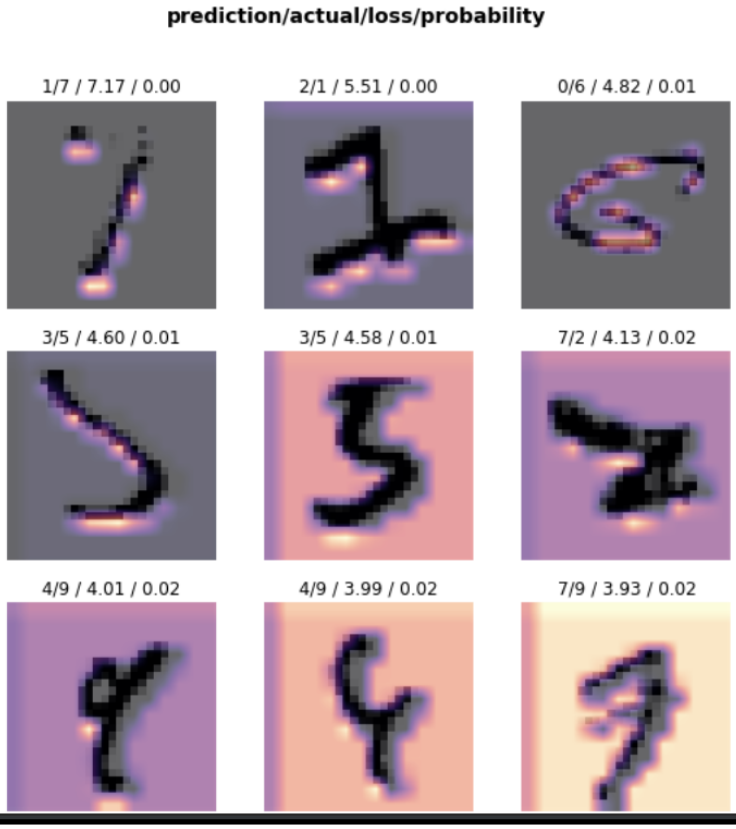
Training steps-

a. This is custom created model.
    i. This model initially had convolution layers nn.Conv2d() (bias=true) , nn.BatchNorm2d and nn.ReLU()
    ii. Not freezing the layers for now as there is no pre-trained models
    iii. Did 3 epochs with learning rate of 0.1 as observed with learn.lr_find()
    iv. Seeing the accuracy is just 98.53 %.
b. Now used the fastai provide conv_layer() which has conv2d(bias !=true), ReLU and BatchNorm2d().
    i. Again not freezing any layers.
    ii. Did 10 epochs and max_lr of 0.1
    iii. Got 99.11% accuracy.
c. Improved the model further by inserting fastai res_block() which introduces additional layers of conv2d but no shape change.
    i. Ran 12 epochs with max_lr = 0.1
    ii. Accuracy = 99.35%
    iii. Ran 12 epochs with max_lr = 0.05
    iv. Accuracy = 99.54%
    v. This is the best accuracy reported in the referred lesson7 of fasti ai course-v3

```
In [51]: interp = ClassificationInterpretation.from_learner(learn)

In [54]: interp.plot_confusion_matrix()
```




```
In [55]: interp.top_losses(k=10)

Out[55]: (tensor([7.1709, 5.5136, 4.8226, 4.5976, 4.5841, 4.1312, 4.0130, 3.9881, 3.9325,
                  3.7987]),
          tensor([9635, 5453, 1582, 8207, 8841, 7384, 4711, 4221, 4835, 1406]))

In [56]: interp.plot_top_losses(9, figsize=(9,9))
```



Observation

a. Most difficult task is to get the data in the right format for computation.

i. Had great difficulty and unable to convert the csv obtained arrays of image pixels into data and thus databunch.
ii. Validation loss and training loss are not decreasing linearly for a complex and high parameter model.
iii. With more epochs, the learned seems to achieve minima.

Concepts explored -

a. Pytorch nn module
   i. nn.Sequential
   ii. nn.BatchNorm2d
   iii. nn.Conv2d
b. Fastai convolution library
   i. Conv_layer - convolution2d with bias=false
   ii. Res_block - for n x n, 2-level convolution
c. Convolution has below parameters in general
   i. Kernel_size
   ii. Stride
   iii. Padding
   iv. Grouping
      1. For group = 1, the convolution layer parameter for ni = 8 and nf = 16
         = 8 * 16 * (3 x 3 (kernel_size)) + 16 (bias==true)
   v. Dilation
   vi. Bias
d. Deep residual learning where extra same input-output dim layer is added with a direct connection from input to output with a adder
   i. Output = input + (conv2(conv1(input))
   ii. This is called ResBlock

Library used - fastai.vision, pytorch- torch.nn

Conclusion:

a. Explored the Hello-World problem of deep learning on MNIST dataset with hand crafted sequential convolution models which resemble resent.

Future work:

a. Use resnet34 or resnet50 by making the data 3 channel.
b. How to use fastai to load arrays in imageList.