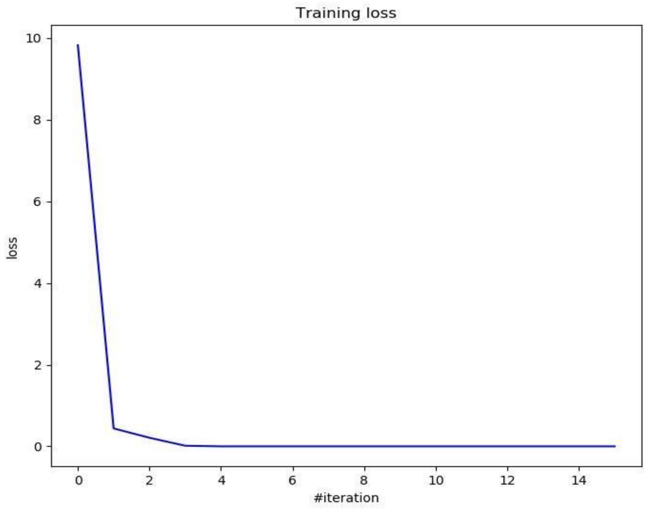
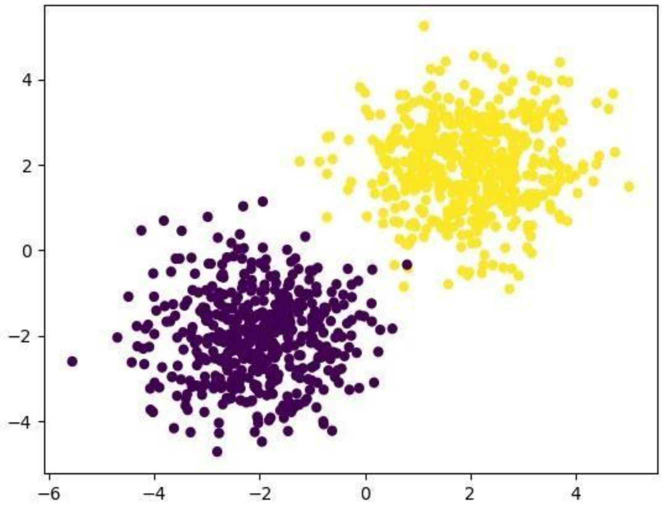
Toy example

There was 2 examples for test:.

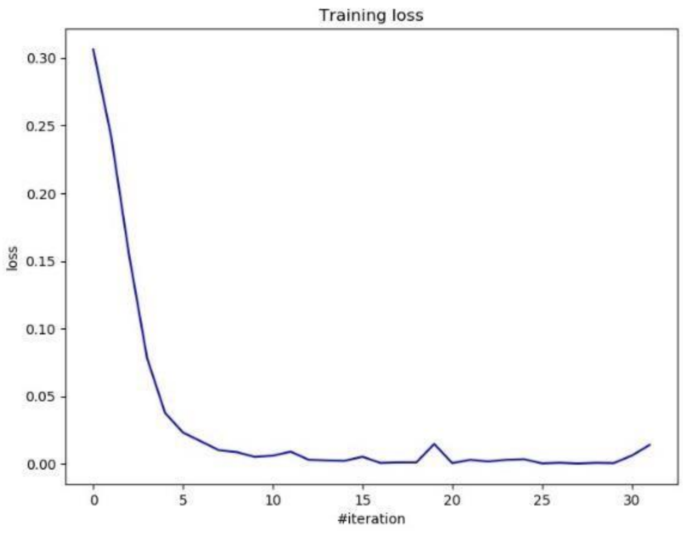
```
net = seq.Sequential() net.add(linear.Linear(2, 4)) net.add(r.ReLU())
net.add(linear.Linear(4, 2)) net.add(softplus.SoftPlus())

criterion = nllu.ClassNLLCriterionUnstable()
```

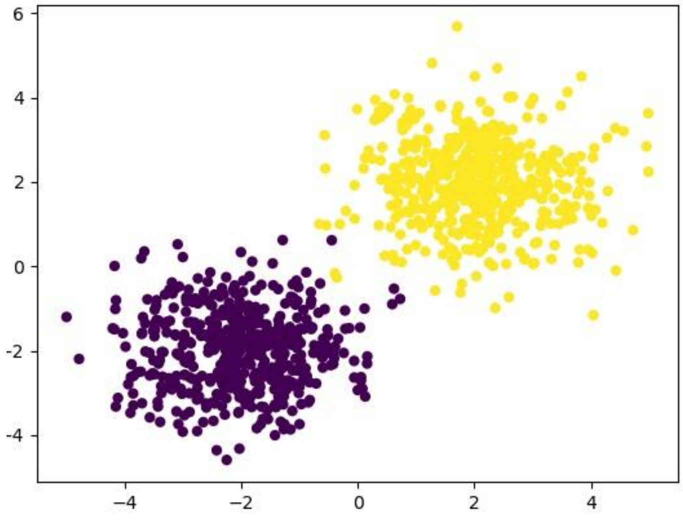
Plots:





```
net = seq.Sequential() net.add(linear.Linear(2, 2))
net.add(softMax.SoftMax())

criterion = mse.MSECriterion()
```





Digit classification

There was main.py file, where networks run. In Networks.py they are written.

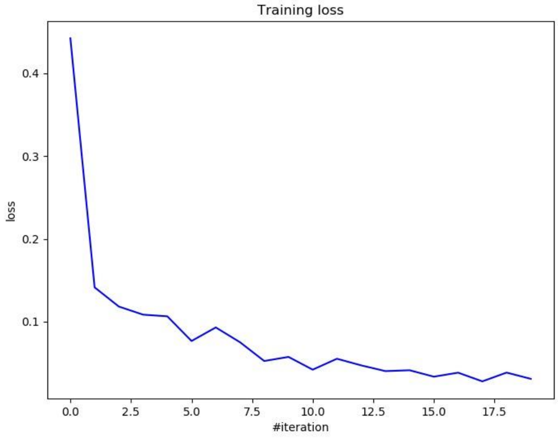
One-hot encode the labels first. (in Dataset.py)

```
from sklearn.preprocessing import OneHotEncoder

onehot_encoder = OneHotEncoder(sparse=False)
train_labels =
train_labels.reshape(len(train_labels), 1)
train_labels =
onehot_encoder.fit_transform(train_labels)
```

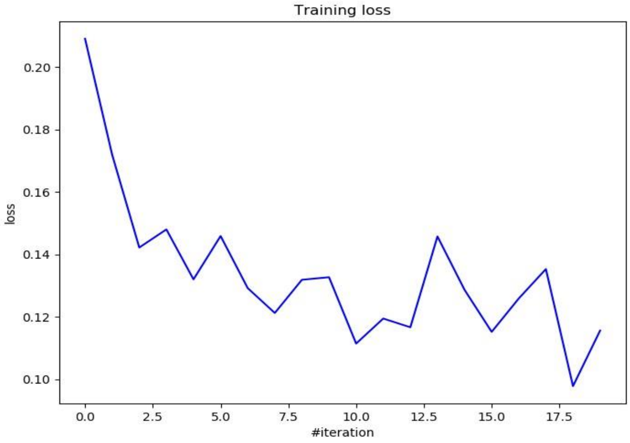
Compare ReLU, ELU, LeakyReLU, SoftPlus activation functions. You would better pick the best optimizer params for each of them, but it is overkill for now. Use an architecture of your choice for the comparison. (in networks.py)

```
    ReLU_net = seq.Sequential()
    ReLU_net.add(linear.Linear(data_size, 100))
    ReLU_net.add(r.ReLU())
    ReLU_net.add(linear.Linear(100, 50))
    ReLU_net.add(r.ReLU())
    ReLU_net.add(linear.Linear(50, predict_size))
    ReLU_net.add(softMax.SoftMax())

    ELU_net = seq.Sequential()
    ELU_net.add(linear.Linear(data_size, 35))
    ELU_net.add(elu.ELU())
    ELU_net.add(linear.Linear(35, predict_size))
    ELU_net.add(softMax.SoftMax)

    LeakyReLU_net = seq.Sequential()
    LeakyReLU_net.add(linear.Linear(data_size, 400))
    LeakyReLU_net.add(leaky.LeakyReLU())
    LeakyReLU_net.add(linear.Linear(400, 250))
    LeakyReLU_net.add(leaky.LeakyReLU())
    LeakyReLU_net.add(linear.Linear(250, predict_size))
    LeakyReLU_net.add(leaky.LeakyReLU())
    LeakyReLU_net.add(softMax.SoftMax)

    SoftPlus_net = seq.Sequential()
    SoftPlus_net.add(linear.Linear(data_size,
predict_size))
    SoftPlus_net.add(softPlus.SoftPlus())
    SoftPlus_net.add(softMax.SoftMax)
```

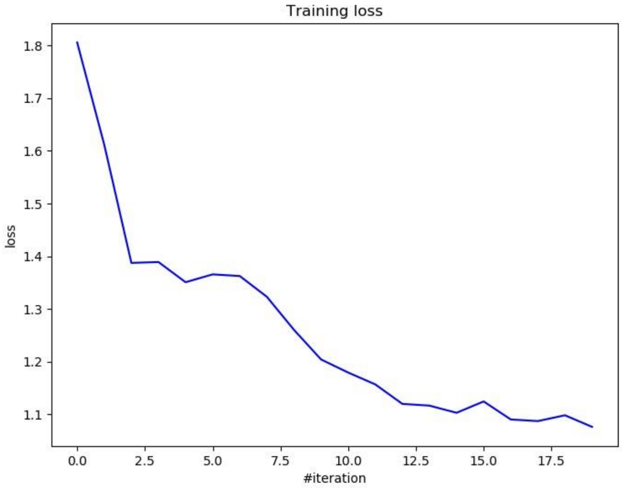
Train data



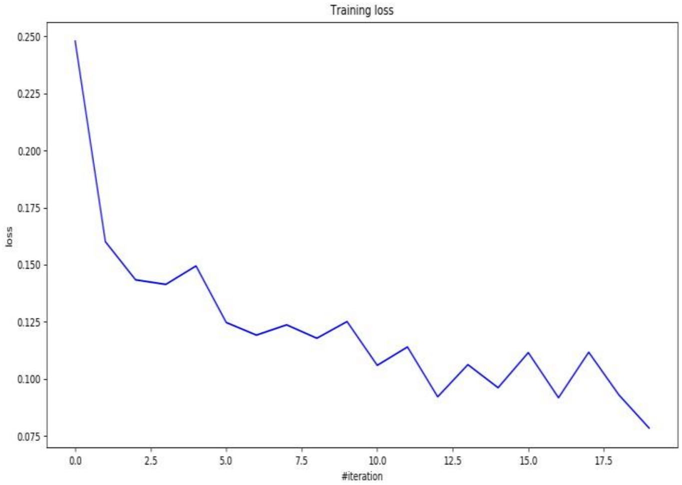
Loss plot for ReLU



Loss plot for ELU



Loss plot for LeakyReLU

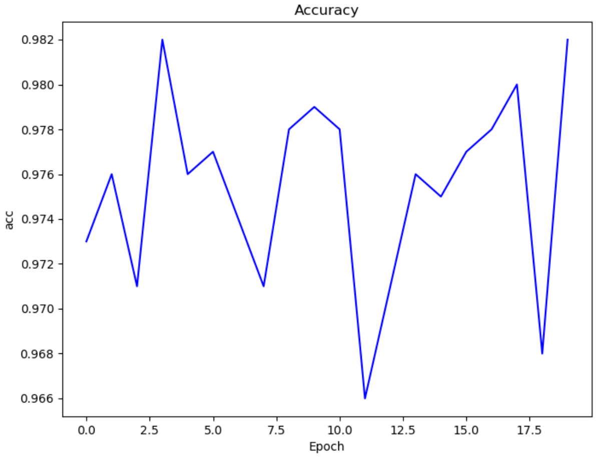


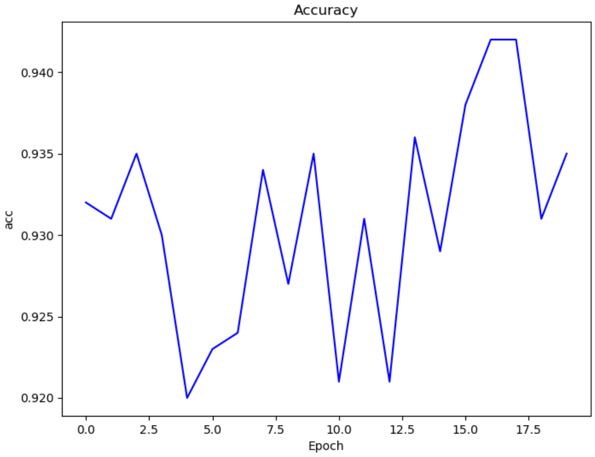
Loss plot for SoftPlus

Test data

ReLU

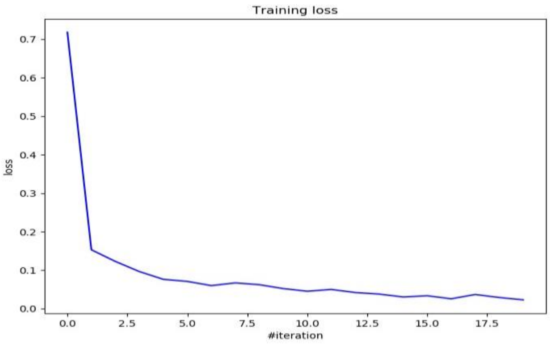
ELU

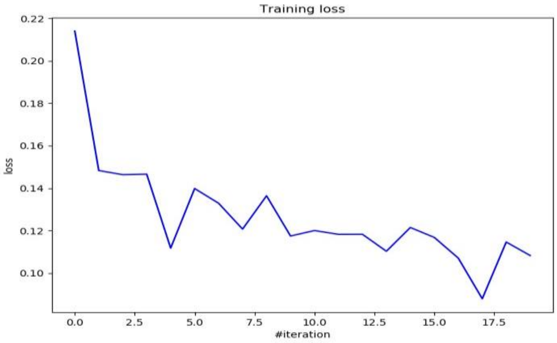




Try inserting `BatchNormalization` (folowed by `ChannelwiseScaling`) between `Linear` module and activation functions.

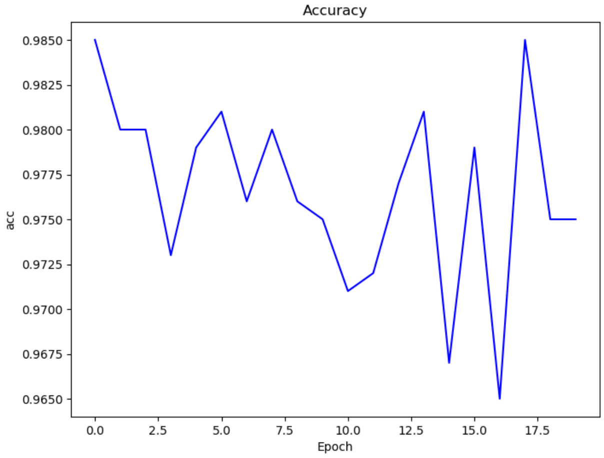
```
ReLU_net = seq.Sequential()
ReLU_net.add(linear.Linear(data_size, 100))
ReLU_net.add(batch.BatchNormalization(0.3))
ReLU_net.add(batch.ChannelwiseScaling(100))
ReLU_net.add(r.ReLU())
ReLU_net.add(linear.Linear(100, predict_size))
ReLU_net.add(softMax.SoftMax())

ELU_net = seq.Sequential()
ELU_net.add(linear.Linear(data_size, predict_size))
ELU_net.add(batch.BatchNormalization())
ELU_net.add(batch.ChannelwiseScaling(predict_size))
ELU_net.add(elu.ELU())
ELU_net.add(softMax.SoftMax())
```



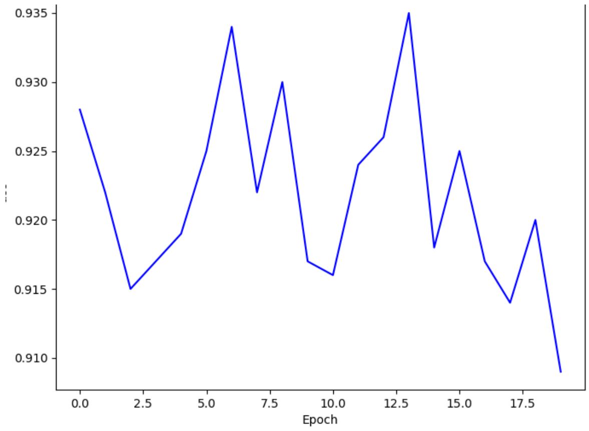


Train data

ReLU

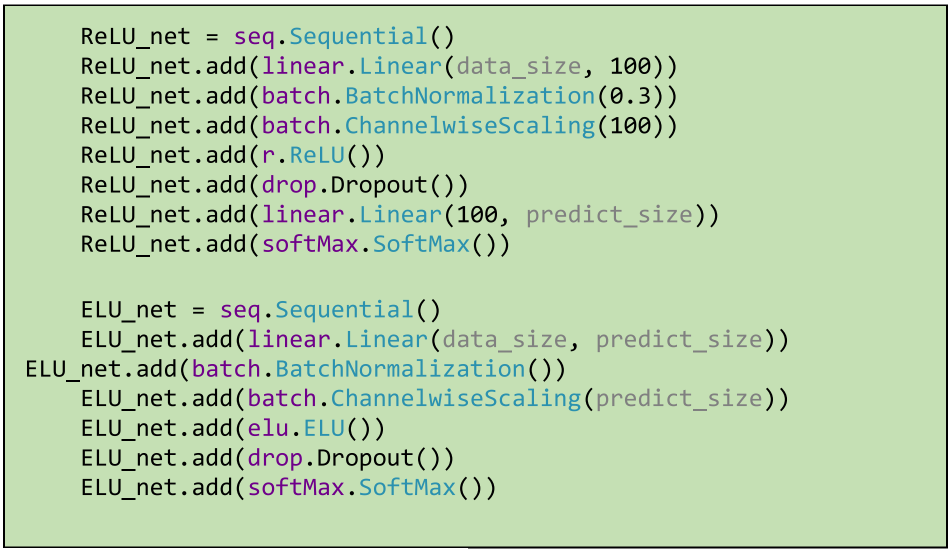


ELU

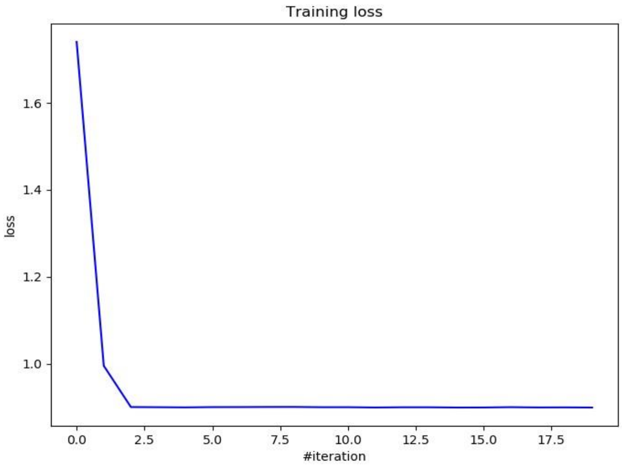


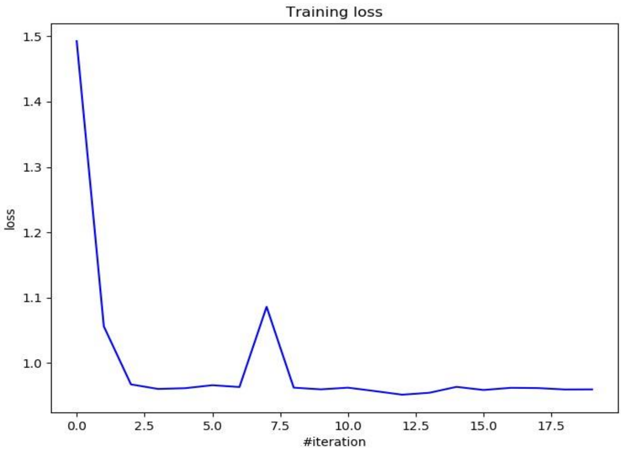
Dropout

Train data



```
    ReLU_net = seq.Sequential()
    ReLU_net.add(linear.Linear(data_size, 100))
    ReLU_net.add(batch.BatchNormalization(0.3))
    ReLU_net.add(batch.ChannelwiseScaling(100))
    ReLU_net.add(r.ReLU())
    ReLU_net.add(drop.Dropout())
    ReLU_net.add(linear.Linear(100, predict_size))
    ReLU_net.add(softMax.SoftMax())

    ELU_net = seq.Sequential()
    ELU_net.add(linear.Linear(data_size, predict_size))
ELU_net.add(batch.BatchNormalization())
    ELU_net.add(batch.ChannelwiseScaling(predict_size))
    ELU_net.add(elu.ELU())
    ELU_net.add(drop.Dropout())
    ELU_net.add(softMax.SoftMax())
```



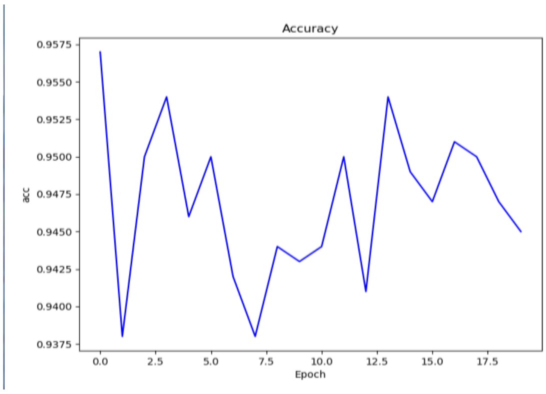


Loss plot for ReLU

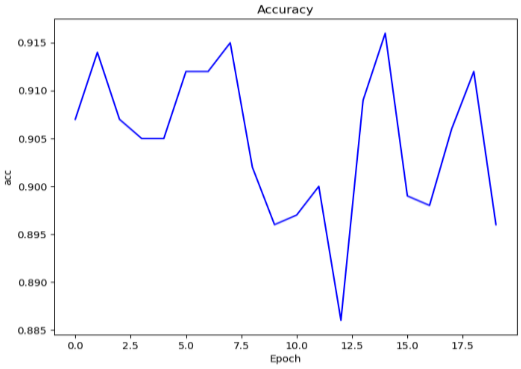
Loss plot for ELU

Test data

ReLU



ELU


Accuracy
acc
Epoch