JADE: Java Deep Learning Library

Elvin M. Munoz V.

Contents

1 Layers implemented **2**

1.1 Linear . . . 2

1.1.1 Example . . . 2

1.2 Conv1d . . . 2

1.2.1 Example . . . 3

1.3 Conv2d . . . 3

1.3.1 Example . . . 3

1.4 MaxPool1d . . . 4

1.4.1 Example . . . 4

1.5 MaxPool2d . . . 4

1.5.1 Example . . . 4

1.6 BatchNorm1d . . . 4

1.6.1 Example . . . 4

1.7 BatchNorm2d . . . 4

1.7.1 Example . . . 4

1.8 Dropout1d . . . 5

1.8.1 Example . . . 5

1.9 Dropout2d . . . 5

1.9.1 Example . . . 5

1.10 Sigmoid . . . 5

1.10.1 Example . . . 5

1.11 Tanh . . . 5

1.11.1 Example . . . 5

1.12 ReLU . . . 6

1.12.1 Example . . . 7

1.13 LeakyReLU . . . 7

1.13.1 Example . . . 7

1.14 Softmax . . . 7

1.14.1 Example . . . 7

1.15 LogSoftmax . . . 8

1.15.1 Example . . . 8

1.16 SiLU . . . 8

1.16.1 Example . . . 8

1.17 GELU . . . 8

1.17.1 Example 9
1.18 Sequential 9
1.18.1 Example 10

2 Loss functions 10
2.1 MSELoss 10
2.1.1 Example 10
2.2 CrossEntropyLoss 10
2.2.1 Example 10
2.3 NLLLoss 10
2.3.1 Example 10

3 Optimizers 11
3.1 SGD 11
3.1.1 Example 11
3.2 Adam 11
3.2.1 Example 11

4 Example codes 11
4.1 Linear Regression 11
4.2 MLP Classification 12
4.3 Convolution Classification 13

1 Layers implemented

1.1 Linear

```
    NNModule linear = new Linear(in_features, out_features,
        bias);
```

$$y = xW + b$$

1.1.1 Example

```
    NNModule linear = new Linear(2, 4, false);
```

1.2 Conv1d

```
    NNModule conv1d = new Conv1d(in_channels, out_channels,
        kernel_size, stride, padding, bias);
```

$$O_{ijk} = \sum_{p=1}^{C}\sum_{q=1}^{K} I'_{i,p,Sk+q-P} F_{j,p,q}$$

$$I'_{ijk} = \begin{cases} I_{ijk}, k >= 1 \wedge k <= H \\ 0, otherwise \end{cases}$$

1.2.1 Example

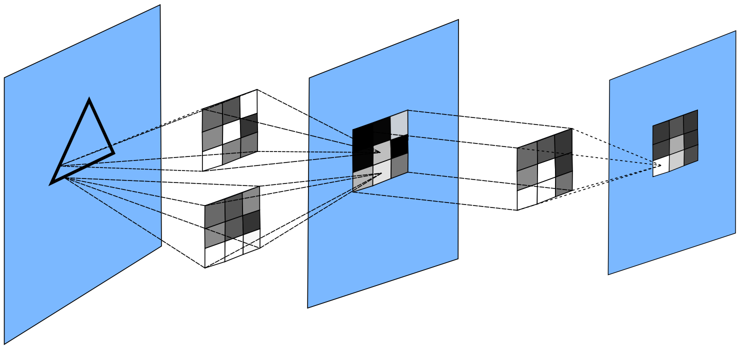
```
    NNModule conv1d = new Conv1d(2, 8, 3, 1, 0, true);
```

1.3 Conv2d

```
    NNModule conv2d = new Conv2d(in_channels, out_channels, kernel_size,  bias);
    NNModule conv2d = new Conv2d(in_channels, out_channels, kernel_size, stride, padding, bias);
```

$$O_{ijkl} = \sum_{p=1}^{C}\sum_{q=1}^{K}\sum_{r=1}^{K} I'_{i,p,Sk+q-P,Sl+r-P} F_{j,p,q,r}$$

$$I'_{ijkl} = \begin{cases} I_{ijkl}, k >= 1 \wedge k <= H \wedge l >= 1 \wedge l <= W \\ 0, otherwise \end{cases}$$



1.3.1 Example

```
    NNModule conv2d = new Conv2d(4, 8, 3, true);
    NNModule conv2d = new Conv2d(4, 8, 4, 1, 0, true);
    NNModule conv2d = new Conv2d(4, 8, new int[] {4, 4} , new int[] {1, 1}, new int[] {0, 0}, true);
```

1.4 MaxPool1d

```
    NNModule maxpool1d = new MaxPool1d(kernel_size);
```

$$O_{ijk} = \max\{I_{i,j,Kk+p}, p >= 0 \land p <= K\}$$

1.4.1 Example

```
    NNModule maxpool1d = new MaxPool1d(4);
```

1.5 MaxPool2d

```
    NNModule maxpool2d = new MaxPool2d(kernel_size);
```

$$O_{ijkl} = \max\{I_{i,j,Kk+p,Kl+q}, p >= 0 \land p <= K \land q >= 0 \land q <= K\}$$

1.5.1 Example

```
    NNModule maxpool2d = new MaxPool2d(2);
    NNModule maxpool2d = new MaxPool2d(new int[] {2, 2});
```

1.6 BatchNorm1d

```
    NNModule batchnorm1d = new BatchNorm1d(numFeatures,
        momentum);
    NNModule batchnorm1d = new BatchNorm1d(numFeatures);
```

$$O_{ijk} = \frac{I_{ijk} - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}} \gamma_j + \beta_j$$

1.6.1 Example

```
    NNModule batchnorm1d = new BatchNorm1d(16, 0.2);
    NNModule batchnorm1d = new BatchNorm1d(16);
```

1.7 BatchNorm2d

```
    NNModule batchnorm2d = new BatchNorm2d(numFeatures,
        momentum);
    NNModule batchnorm2d = new BatchNorm2d(numFeatures);
```

$$O_{ijkl} = \frac{I_{ijkl} - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}} \gamma_j + \beta_j$$

1.7.1 Example

```
    NNModule batchnorm2d = new BatchNorm2d(16, 0.2);
    NNModule batchnorm2d = new BatchNorm2d(16);
```

1.8 Dropout1d

```
    NNModule dropout1d = new Dropout1d(p);
```

$$O_{ijk} = \begin{cases} 0 & ; random(0,1) < p \\ \frac{I_{ijk}}{p} & ; otherwise \end{cases}$$

1.8.1 Example

```
    NNModule dropout1d = new Dropout1d(0.5);
```

1.9 Dropout2d

```
    NNModule dropout2d = new Dropout2d(p);
```

$$O_{ijkl} = \begin{cases} 0 & ; random(0,1) < p \\ \frac{I_{ijkl}}{p} & ; otherwise \end{cases}$$

1.9.1 Example

```
    NNModule dropout2d = new Dropout2d(0.5);
```

1.10 Sigmoid

```
    NNModule sigmoid = new Sigmoid();
```

$$y = \sigma(x) = \frac{1}{1 + e^{-x}}$$

1.10.1 Example

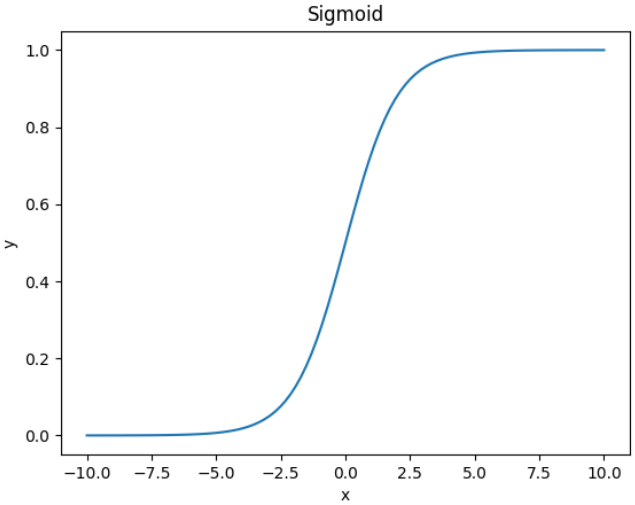
```
    NNModule sigmoid = new Sigmoid();
```

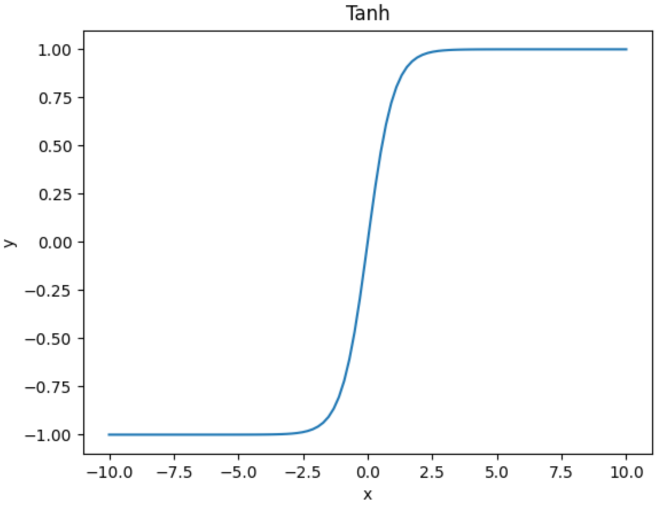
1.11 Tanh

```
    NNModule tanh = new Tanh();
```

$$y = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

1.11.1 Example



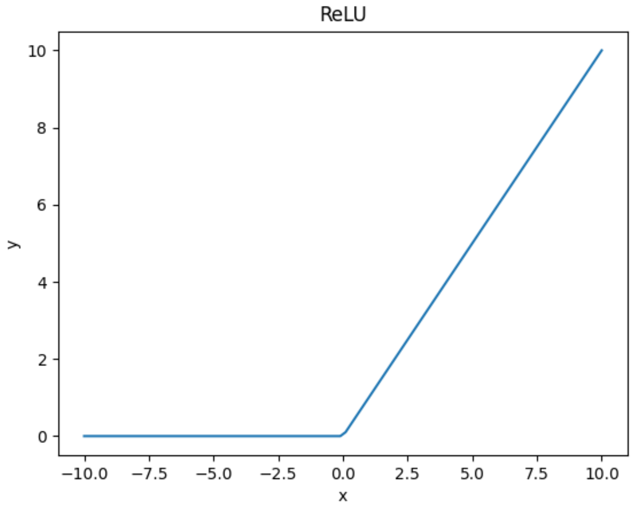


```
        NNModule tanh = new Tanh();
```

1.12 ReLU

```
        NNModule relu = new ReLU();
```

$$y = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$$



1.12.1 Example

```
    NNModule relu = new ReLU();
```

1.13 LeakyReLU

```
    NNModule leakyrelu = new LeakyReLU(alpha);
```

$$y = \begin{cases} \alpha x & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$$

1.13.1 Example

```
    NNModule leakyrelu = new LeakyReLU(0.01);
```

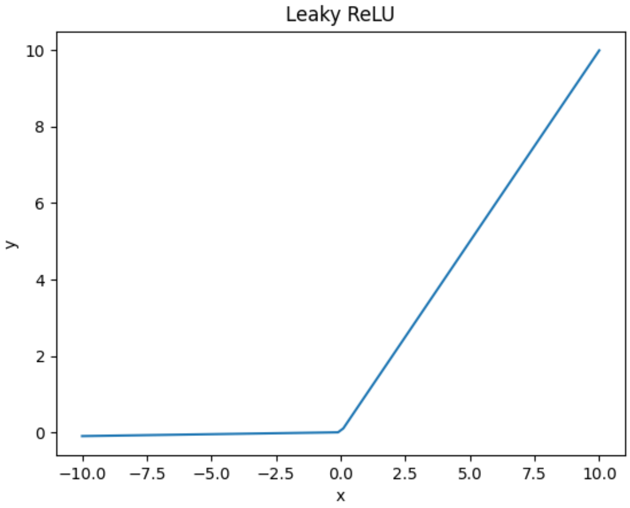
1.14 Softmax

```
    NNModule softmax = new Softmax();
```

$$y_i = \frac{e^{x_i}}{\sum_k e^{x_k}}$$

1.14.1 Example

```
    NNModule softmax = new Softmax();
```



1.15 LogSoftmax

```
    NNModule logsoftmax = new LogSoftmax();
```

$$y_i = \log \frac{e^{x_i}}{\sum_k e^{x_k}}$$

1.15.1 Example

```
    NNModule logsoftmax = new LogSoftmax();
```

1.16 SiLU

```
    NNModule silu = new SiLU();
```

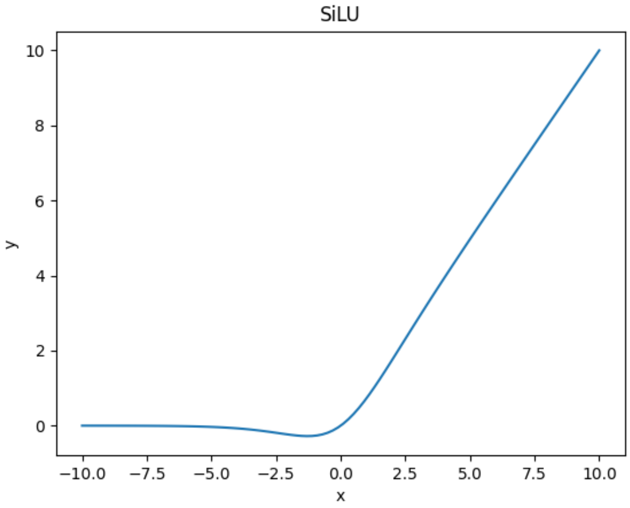
$$y = x\sigma(x) = \frac{x}{1 + e^{-x}}$$

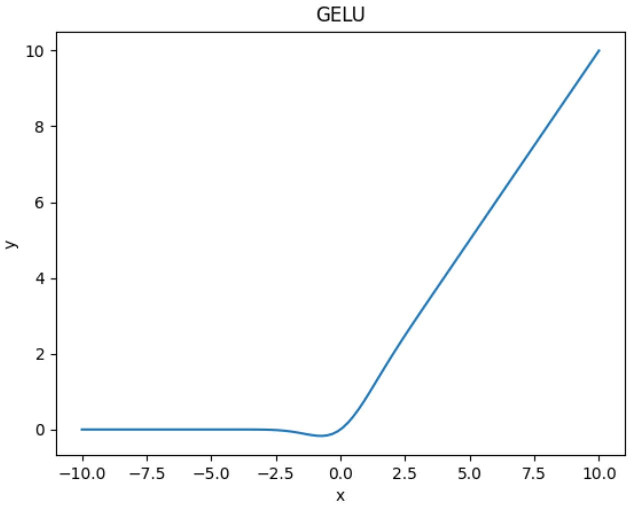
1.16.1 Example

```
    NNModule silu = new SiLU();
```

1.17 GELU

```
    NNModule gelu = new GELU();
```


SiLU


GELU

$$y \approx x\sigma(1.702x) = \frac{x}{1 + e^{-1.702x}}$$

1.17.1 Example

```
    NNModule gelu = new GELU();
```

1.18 Sequential

```
    NNModule sequential = new Sequential();
```

1.18.1 Example

```
    NNModule sequential = new Sequential();
    model.add_module((NNModule) new Conv2d(1, 8, 3, true));
    model.add_module((NNModule) new Sigmoid());
    model.add_module((NNModule) new MaxPool2d(2));
    model.add_module((NNModule) new Conv2d(8, 16, 3, true));
    model.add_module((NNModule) new Sigmoid());
    model.add_module((NNModule) new Flatten());
    model.add_module((NNModule) new Linear(16, 10, true));
```

2 Loss functions

2.1 MSELoss

```
    Loss loss_fn = new MSELoss();
```

$$\mathcal{L}(y,t) = \frac{1}{N}\sum_{i=1}^{N}(y_i - t_i)^2$$

2.1.1 Example

```
    Loss loss_fn = new MSELoss();
```

2.2 CrossEntropyLoss

```
    Loss loss_fn = new CrossEntropyLoss();
```

$$\mathcal{L}(y,t) = -\sum_{i=1}^{N} t_i \log(y_i)$$

2.2.1 Example

```
    Loss loss_fn = new CrossEntropyLoss();
```

2.3 NLLLoss

```
    Loss loss_fn = new NLLLoss();
```

$$\mathcal{L}(y,t) = -\sum_{i=1}^{N} t_i y_i$$

2.3.1 Example

```
    Loss loss_fn = new NLLLoss();
```

3 Optimizers

3.1 SGD

```
    Optimizer optim = new SGD(parameters, hyperparameters);
```

3.1.1 Example

```
    HashMap<String, Float> hyperparameters = new HashMap<>()
        ;
    hyperparameters.put("lr", 0.01f);
    hyperparameters.put("momentum", 0.9f);
    Optimizer optim = new SGD(model.parameters(),
        hyperparameters);
```

3.2 Adam

```
    Optimizer optim = new Adam(parameters, hyperparameters);
```

3.2.1 Example

```
    HashMap<String, Float> hyperparameters = new HashMap<>()
        ;
    hyperparameters.put("lr", 0.01f);
    Optimizer optim = new Adam(model.parameters(),
        hyperparameters);
```

4 Example codes

4.1 Linear Regression

```
import com.nn.*;
import com.optim.*;
import com.data.*;
import java.util.*;

public class SampleLinearRegression {
    public static void main(String args[]) {
        float[] X = new float[] { 1, 2, 3, 4, 5 };
        float[] y = new float[] { 7, 9, 11, 13, 15 };

        Tensor x_train = new Tensor(new int[] { 5, 1 }, X);
        Tensor y_train = new Tensor(new int[] { 5, 1 }, y);

        NNModule seq = new Sequential();
        seq.add_module((NNModule) new Linear(1, 1, true));

        Loss loss_fn = new MSELoss();
```

```java
        Map<String, Float> optim_params = new HashMap<String, Float>();
        optim_params.put("lr", 0.01f);
        Optimizer optim = new SGD(seq.parameters(), optim_params);

        for (int epoch = 0; epoch < 1000; epoch++) {
            Tensor o = seq.forward(x_train);
            Tensor loss = loss_fn.criterion(o, y_train);
            if (epoch % 100 == 0) {
                System.out.println(epoch + ": " + loss);
            }
            optim.zero_grad();
            loss.backward();
            optim.step();
        }

        for (Tensor param : seq.parameters()) {
            param.print();
        }
    }
}
```

4.2 MLP Classification

```java
import com.nn.*;
import com.optim.*;
import com.data.*;
import com.utils.*;
import java.util.*;

import java.nio.file.Paths;

public class SampleMLPClassification {
        public static void main(String[] args) {
                Sequential model = new Sequential();
                model.add_module(new Flatten());
                model.add_module(new Linear(64, 32, true));
                model.add_module(new Tanh());
                model.add_module(new Linear(32, 16, true));
                model.add_module(new Tanh());
                model.add_module(new Linear(16, 10, true));

                String path_to_digits = "./data/DIGITS/";

                Tensor x_train = Misc.loadTensor(Paths.get(path_to_digits,
                                "x_train_digits.bin").toString());
                Tensor y_train = Misc.loadTensor(Paths.get(
                    path_to_digits,
                                "y_train_digits.bin").
                                    toString());
                Tensor x_test = Misc.loadTensor(Paths.get(
                    path_to_digits,
                                "x_test_digits.bin").
                                    toString());
                Tensor y_test = Misc.loadTensor(Paths.get(
                    path_to_digits,
                                "y_test_digits.bin").
                                    toString());

                x_train = x_train.div(new Tensor(16.0f));
                x_test = x_test.div(new Tensor(16.0f));

                Dataset train_ds = new TensorDataset(x_train
                    , y_train);
                DataLoader train_dl = new DataLoader(
                    train_ds, 32, true);

                Dataset test_ds = new TensorDataset(x_test,
                    y_test);
                DataLoader test_dl = new DataLoader(test_ds,
                     32, true);

                HashMap<String, Float> hyperparams = new
                    HashMap<String, Float>();
                hyperparams.put("lr", 0.01f);
                hyperparams.put("momentum", 0.9f);
                Optimizer optim = new Adam(model.parameters
                    (), hyperparams);
                Loss loss_fn = new CrossEntropyLoss();

                Misc.train(model, train_dl, test_dl, optim,
                    loss_fn, 10);
        }
}
```

4.3 Convolution Classification

```
import com.data.Tensor;
import com.utils.*;
import com.optim.*;
import com.nn.*;

import java.nio.file.Paths;
import java.util.HashMap;
import java.util.Map;

public class SampleConvClassification {
        public static void main(String args[]) {
                Sequential model = new Sequential();

                model.add_module((NNModule) new Conv2d(1, 8,
                     3, true));
                model.add_module((NNModule) new Sigmoid());
                model.add_module((NNModule) new MaxPool2d(2)
                    );
                model.add_module((NNModule) new Conv2d(8,
                    16, 3, true));
                model.add_module((NNModule) new Sigmoid());
                model.add_module((NNModule) new Flatten());
                model.add_module((NNModule) new Linear(16,
                    10, true));

                String path_to_digits = "./data/DIGITS/";
                DataLoader[] dl = com.vision.Datasets.
                    loadDIGITS(path_to_digits, 64);

                HashMap<String, Float> hyperparams = new
                    HashMap<String, Float>();
                hyperparams.put("lr", 0.01f);
                hyperparams.put("momentum", 0.9f);
                Optimizer optim = new Adam(model.parameters
                    (), hyperparams);
                Loss loss_fn = new CrossEntropyLoss();

                Misc.train(model, dl[0], dl[1], optim,
                    loss_fn, 50);
        }

}
```