기초 인공지능
Assignment04 – Neural Network

20191621 이민영

1. 직접 구현한 MLP model의 layer 구성에 대해 설명하세요.

nn.Sequential에 nn.Linear와 self.activation_func을 이용하여 구성하였다.
nn.Linear는 32*32*3, 32*16*3을 인자로 주어서 32*32*3의 input size 32*32*3를 32*16*3으로 출력될 수 있도록 하였다. 이후, activation function을 통해서 출력 값을 다음 레이어로 보내주었다.

```
nn.Linear(32*16*3,16*16*3),self.activation_func,
nn.Linear(16*16*3,16*8*3),self.activation_func,
nn.Linear(16*8*3,8*8*3),self.activation_func,
nn.Linear(8*8*3,8*4*3),self.activation_func,
nn.Linear(8*4*3,10)
```

위와 같이 맨처음 구현과 비슷한 형식으로 구현하였다.
이후, torch.flatten함수를 사용해서 정의한 flatten함수를 이용해서 forward함수에서 flatten 해주었다.

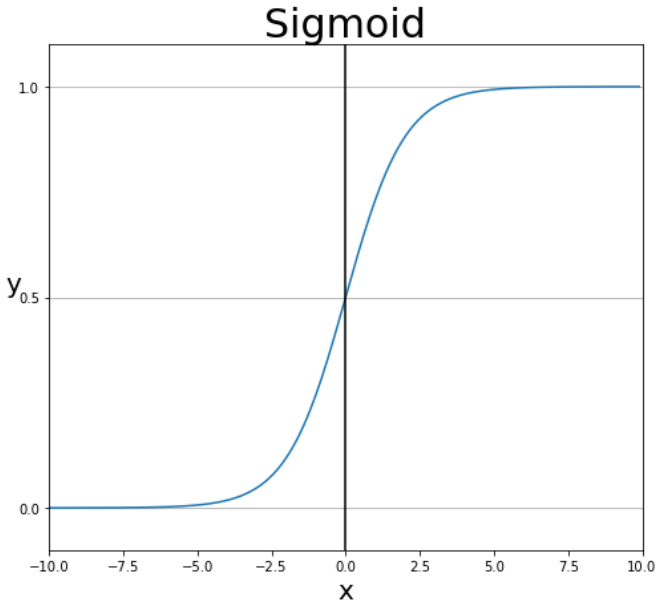
2. 직접 구현한 CNN model의 layer 구성에 대해 설명하세요.

nn.Conv2d, nn.MaxPool2d, nn.Linear를 이용해서 구현하였다.
32*32*3을 30*30*12로 만들기 위해서 입력 채널을 3, 출력 채널 수를 12로 두었고, 커널 사이즈를 5로 두었다. 이 때, 32를 30으로 만들어야하기 때문에 padding을 1로 두었다.
마찬가지로 28*28*12로 만들기 위해서 입력 채널과 출력 채널 수를 12로 두었고, 커널 사이즈와 padding을 동일하게 주었다. 이후, 14*14*12로 만들기 위해서 nn.MaxPool2d(2)를 설정하였다.
위와 동일하게 12*12*24->10*10*24를 위해서
nn.Conv2d(12,24,5, padding = 1), nn.Conv2d(24,24,5,padding = 1)을 하고, 마지막 5*5*24로 pool위해 nn.MaxPool2d(2)를 self.pool2에 대입하였다.
fc층을 3개로 두어 nn.Linear를 통해 대입하였다.
이후, forward에서 activation과 pool, pool2를 이용해서 구현하였다. conv2 이후에 pool을 적용하고 conv4 이후에 pool2이 적용될 수 있도록 하였으며 flatten 함수와 fc층을 사용하였다.

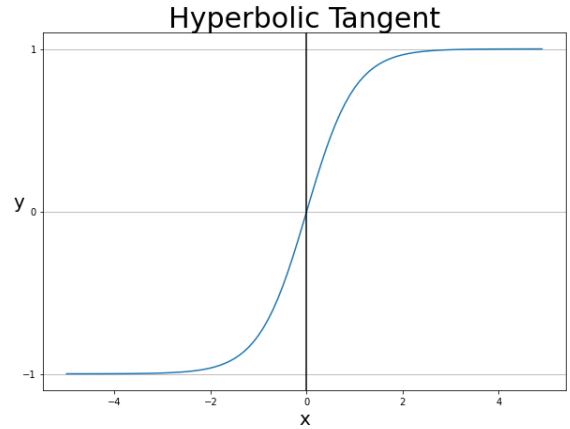
3. 선택한 Activation function에 대해 설명하세요.

각각의 Activation function에 대해서 살펴보면,

우선 시그모이드 함수는 0에서 1 사이의 함수이며, 값이 들어왔을 때 0과 1 사이의 값을 반환하게 되며, 시그모이드 함수를 활성화함수로 사용하면 0과 1에 가까운 값을 통해서 이진 분류를 할 수 있다. 매끄러운 곡선을 가지기 때문에 기울기가 급격하게 변해서 발산하는 gradient exploding이 발생하지 않는다는 장점이 있다. 하지만 . 시그모이드 함수를 사용했을 때는 출력하는 값의 범위가 0에서 1 사이이기 때문에 layer를 거치게 되면 값이 매우 작아져서 vanishing gradient 현상이 발생하게 된다는 단점이 있다.

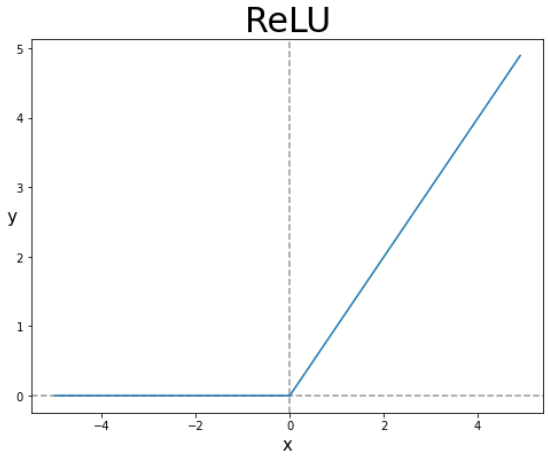


두번째로 tanh 함수는 -1에서 1 사이의 값을 출력하며, 중앙값이 0이다. tanh 함수는 중앙값이 0이기 때문에 경사하강법 사용 시 시그모이드 함수에서 발생하는 편향 이동이 발생하지 않는다는 장점이 있지만, 구간이 많이 크지는 않기 때문에 gradient vanishing 증상이 여전히 존재한다는 단점이 있다.

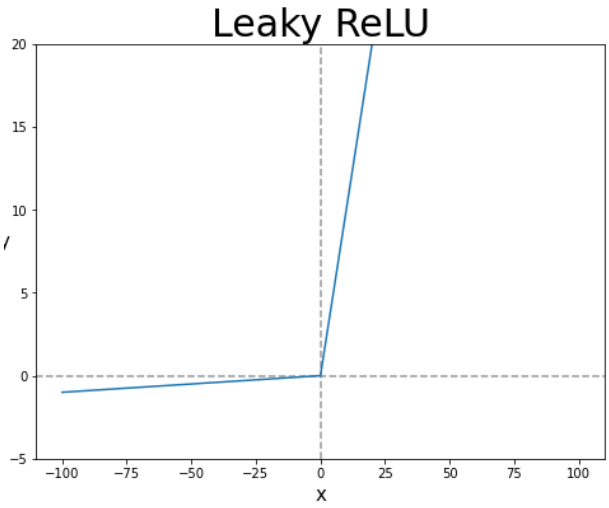


세번째로 ReLU 함수는 가장 많이 사용되는 활성화 함수로, 값이 양수이면 자기 자신을 반환하고, 음수이면 0을 반환하는 함수이다. 이번 과제의 활성화 함수로 ReLU 함수를 선택하였다. ReLU함

수는 양수인 경우에 자기 자신을 그대로 출력하기 때문에 특정 양수 값에 수렴하지 않고 출력 값의 범위가 넓기 때문에 vanishing gradient문제가 발생하지 않는다. 또한 식이 단순하기 때문에 다른 활성화 함수에 비해서 학습 속도가 빠르다는 장점이 있다. 이러한 이유로 이번 과제에 활성화 함수로 선택하였다.



마지막으로 leaky ReLU함수는 음수에 아주 작은 값을 곱해서 dying ReLU를 막는 함수이다. 그러나 음수에서 선형성이 생기게 되고 그로 인해서 복잡한 분류에서 사용할 수 없다는 한계가 생긴다는 단점이 있다. 따라서 음수가 아주 중요한 상황일 때 주로 사용하게 된다.



위의 이러한 특성들로 인해 ReLU를 선택하였고, 실제로 다른 활성화함수에 비해서 더 좋은 결과값을 얻을 수 있었다.

4. MLP Test Accuracy 와 CNN Test Accuracy 캡쳐 화면

```
---------- MLP Test Result ----------------
MLP Test Accuracy: 44.225
```

```
---------- CNN Test Result ----------
CNN Test Accuracy: 54.8
```

5. **Validation Accuracy Graph 캡쳐 화면**