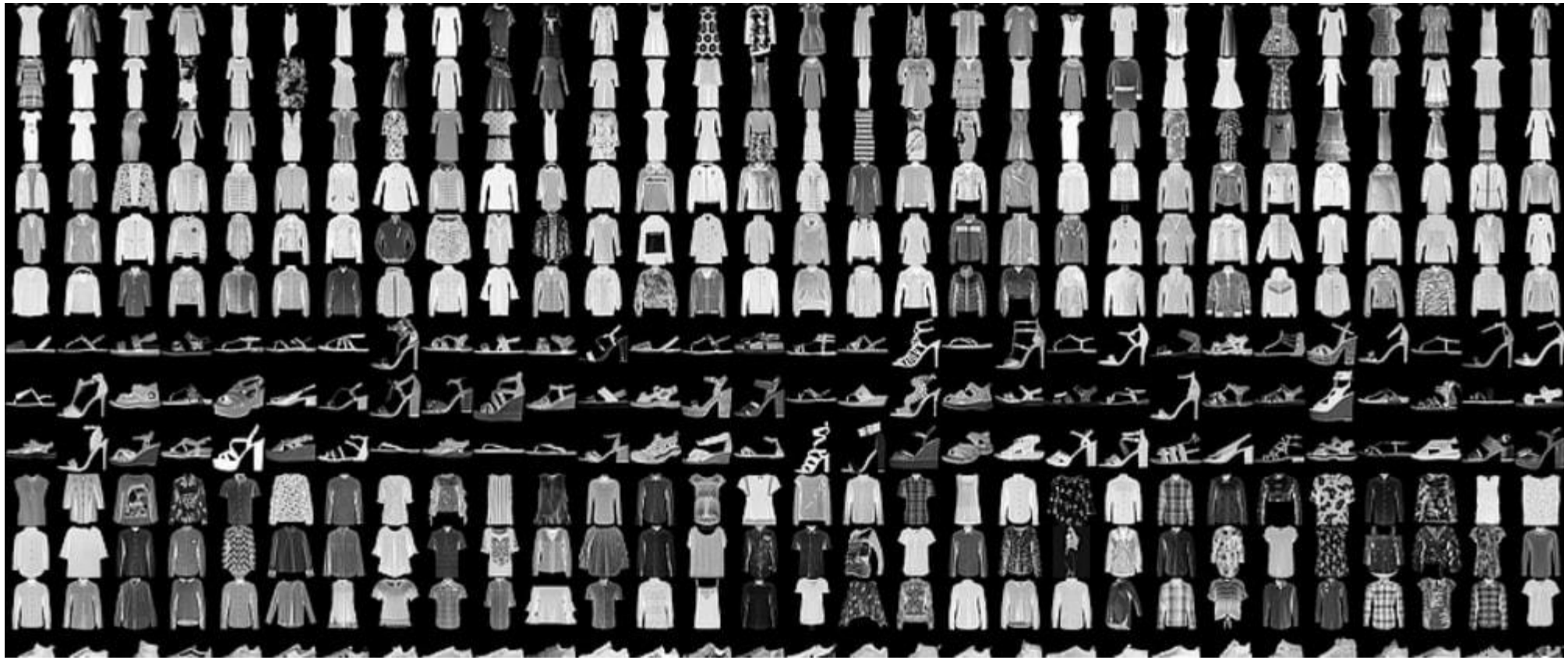
Fashion MNIST

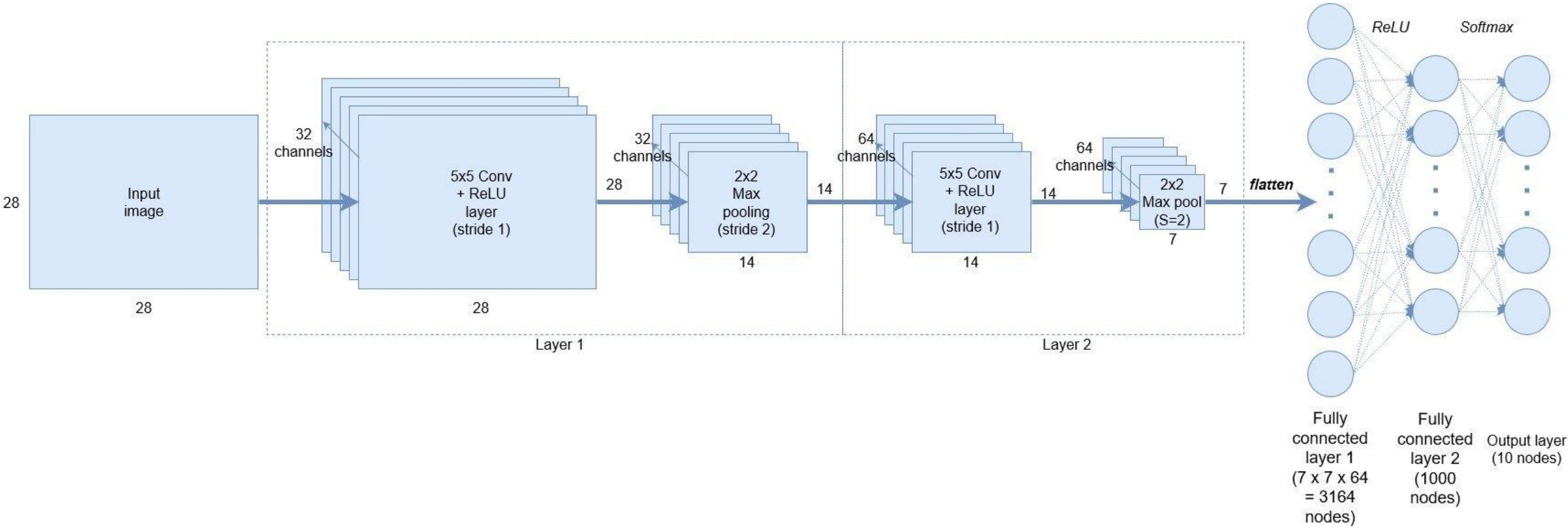
박경규

Fashion MNIST

- Fashion MNIST 데이터셋은 운동화, 셔츠, 샌들과 같은 작은 이미지들의 모음입니다.
- 기본 MNIST 데이터셋과 같이 열 가지로 분류될 수 있는 28×28 픽셀의 이미지 70,000개로 이루어져 있습니다.



CNN 구조



https://jschang.tistory.com/4

데이터 로드

```
train_set = torchvision.datasets.FashionMNIST(
    root = './data/FashionMNIST',
    train = True,
    download = True,
    transform = transforms.Compose([
        transforms.ToTensor() # 데이터를 0에서 255까지 있는 값을 0에서 1사이 값으로 변환
    ])
)
test_set = torchvision.datasets.FashionMNIST(
    root = './data/FashionMNIST',
    train = False,
    download = True,
    transform = transforms.Compose([
        transforms.ToTensor() # 데이터를 0에서 255까지 있는 값을 0에서 1사이 값으로 변환
    ])
)
train_loader = torch.utils.data.DataLoader(train_set, batch_size=batch_size)
test_loader = torch.utils.data.DataLoader(test_set, batch_size=batch_size)
```

https://jschang.tistory.com/4

모델 아키텍처

```
class ConvNet(nn.Module):
    def __init__(self):
        super().__init__()
        self.layer1 = nn.Sequential( # 순차적인 레이어 쌓게 함
            # Convolution + ReLU + max Pool
            nn.Conv2d(in_channels=1, out_channels=32, kernel_size=5, stride=1, padding=2),
            # Wout = (Win - FilterSize + 2*Padding)/Stride + 1
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2)
        )
        self.layer2 = nn.Sequential( # 순차적인 레이어 쌓게 함
            # Convolution + ReLU + max Pool
            nn.Conv2d(in_channels=32, out_channels=64, kernel_size=5, stride=1, padding=2),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2)
        )
        self.dropout = nn.Dropout() # over-fitting 방지
        self.fc1 = nn.Linear(in_features=7*7*64, out_features=1000)
        self.fc2 = nn.Linear(in_features=1000, out_features=10)
```

https://jschang.tistory.com/4

모델 아키텍처

```
    def forward(self, x):
        x = self.layer1(x)
        x = self.layer2(x)
        x = x.reshape(x.size(0), -1)
        x = self.dropout(x) # 오버피팅을 막기 위해 학습 과정시 일부 뉴런을 생략하는 기능
        x = self.fc1(x)
        x = self.fc2(x)
        return x
```

```
model = ConvNet()

critertion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr = learning_rate)
```

https://jschang.tistory.com/4

모델 훈련

```
model = ConvNet()

critertion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr = learning_rate)
```

```
total_step = len(train_loader)
pd_results = []

for epoch in range(epochs):
    for i, (images, labels) in enumerate(train_loader):
        out = model(images) # 자동으로 forward 함수 불러옴
        loss = critertion(out, labels)

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```

https://jschang.tistory.com/4

모델 훈련

```
    total = labels.size(0)
    preds = torch.max(out.data, 1)[1]
    # out은 리스트로 가장 큰 값을 가진 원소의 인덱스가 예측 클래스
    # torch.max의 파라미터 중 1은 1차원의 리스트로 출력한다
    # 이는 labels이 1차원 리스트로 같게 맞춰주기 위함
    correct = (preds==labels).sum().item() # 같은 것은 합해서 맞은 개수 얻음

    if (i+1)%100==0:
        results = OrderedDict()
        results['epoch'] = epoch+1
        results['idx'] = i+1
        results['loss'] = loss.item()
        results['accuracy'] = 100.*correct/total
        pd_results.append(results)
        df = pd.DataFrame.from_dict(pd_results, orient='columns')

        clear_output(wait=True)
        display(df)
```

https://jschang.tistory.com/4

Fashion MNIST 실습

https://jschang.tistory.com/4

Thank you

kgpark88@gmail.com