10-315
Introduction to ML

ML Applications in Practice

Instructor: Pat Virtue

Outline

ML Applications in Practice

- [Model design challenge]
- [ML model cards]
- [Deep learning toolkits (PyTorch)]

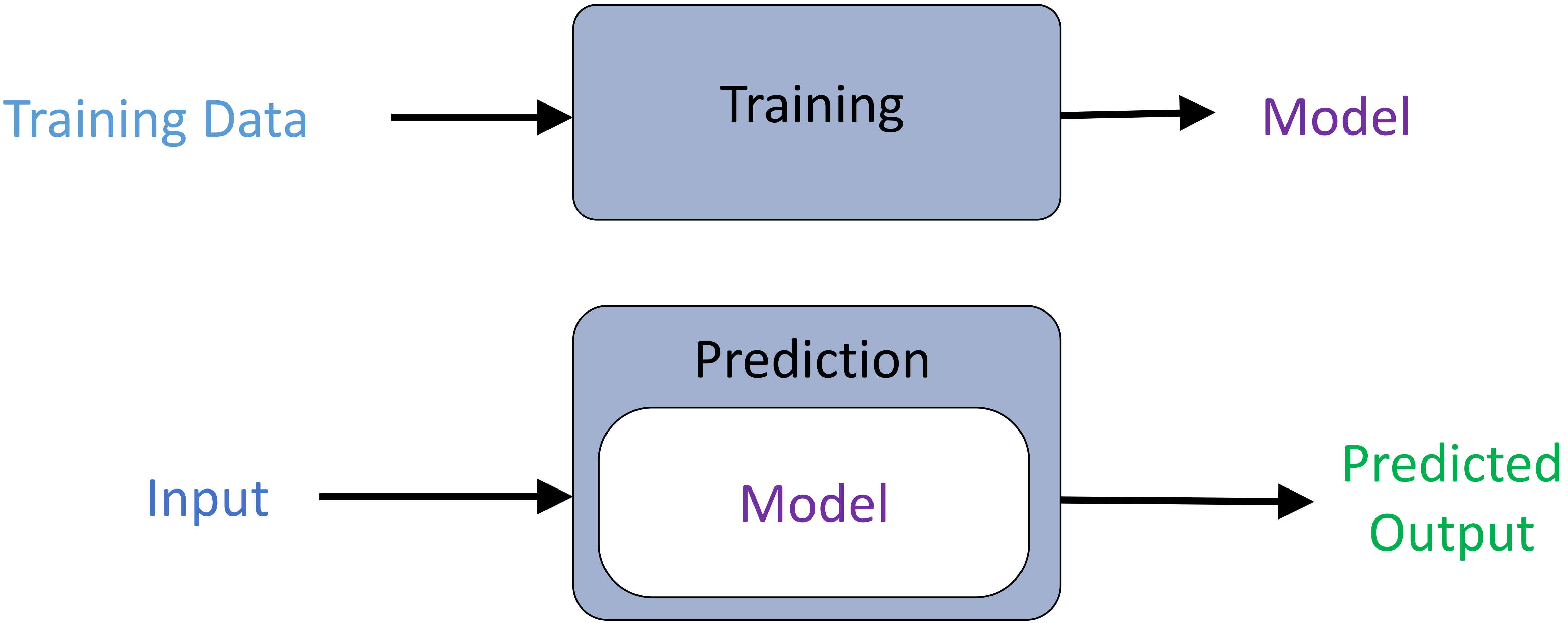
Next time

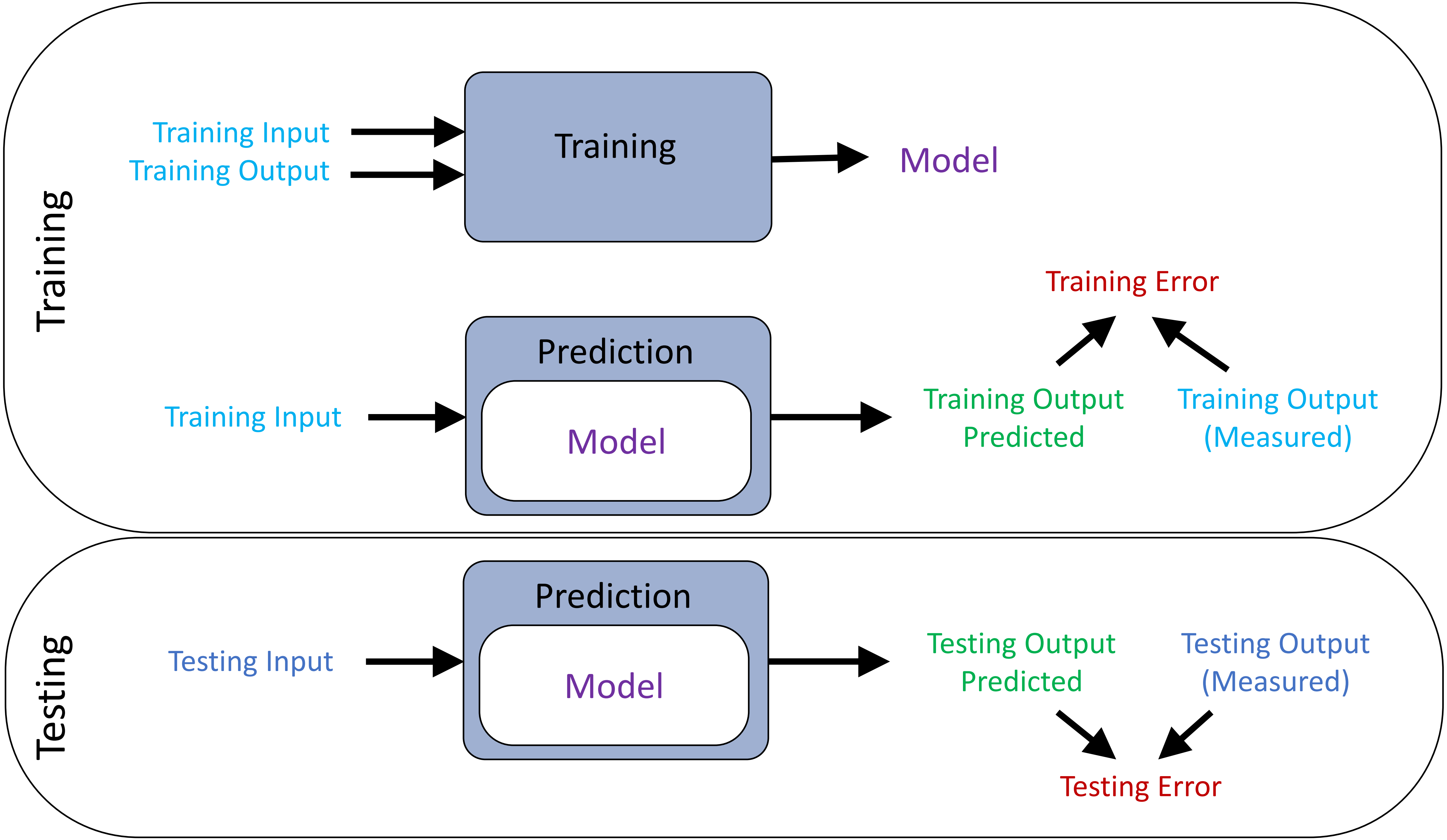
- Neural networks applications for images
- Neural networks applications for language

Model Design

Reminder: Machine Learning Tasks

Using (training) data to learn a model that we'll later use for prediction




Training
Training Input
Training Output
Training
Model
Training Error
Prediction
Training Input
Model
Training Output
Predicted
Training Output
(Measured)
Testing
Prediction
Testing Input
Model
Testing Output
Predicted
Testing Output
(Measured)
Testing Error

AI in the News

News: https://gadgets.ndtv.com/social-networking/news/flemish-scrollers-ai-ml-bot-tool-software-belgium-politicians-distracted-phone-detect-troll-twitter-2480493

Source: Twitter @Flemish Scroller

Gadgets 360 An NDTV venture

HOME GUIDE NEWS REVIEWS FEATURES

AI-Based Bot Detects Politicians Distracted by Phone, Posts Photo on Twitter and Asks Them to Focus

The software searches for phones and then look for distracted politicians during livestreams of parliamentary sessions.

By Edited by Gadgets 360 Newsdesk | Updated: 6 July 2021 16:34 IST

Share on Facebook | Tweet | Snapchat | Share | Reddit | Email | Comment

ML App Design

Distracted Politician Detection

- What is the input/output for the task?

Image credit: Twitter @Flemish Scroller

Input/Output

Distracted Politician Detection

Design the app by connecting smaller input/output ML Tasks

Image →

→ action   sent tweet to name with img

→ name

→ img

→ % distracted

→ bb phone coord [ul.r, ul.c, w, h]

→ bb face

Image credit: Twitter @Flemish Scroller

Input/Output

Distracted Politician Detection

- What is the input/output for the task?
- What is the performance measure?
- What training data do we need?
- What is the intended use?
- What could go wrong?

Image credit: Twitter @Flemish Scroller

ML Model Cards

ML Model Cards

Mitchell, Margaret, et al.

"Model cards for model reporting.“

Proceedings of the conference on fairness, accountability, and transparency. 2019.

ML Model Cards

Model Card

- **Model Details**. Basic information about the model.
  - Person or organization developing model
  - Model date
  - Model version
  - Model type
  - Information about training algorithms, parameters, fairness constraints or other applied approaches, and features
  - Paper or other resource for more information
  - Citation details
  - License
  - Where to send questions or comments about the model
- **Intended Use**. Use cases that were envisioned during development.
  - Primary intended uses
  - Primary intended users
  - Out-of-scope use cases
- **Factors**. Factors could include demographic or phenotypic groups, environmental conditions, technical attributes, or others listed in Section 4.3.
  - Relevant factors
  - Evaluation factors
- **Metrics**. Metrics should be chosen to reflect potential real-world impacts of the model.
  - Model performance measures
  - Decision thresholds
  - Variation approaches
- **Evaluation Data**. Details on the dataset(s) used for the quantitative analyses in the card.
  - Datasets
  - Motivation
  - Preprocessing
- **Training Data**. May not be possible to provide in practice. When possible, this section should mirror Evaluation Data. If such detail is not possible, minimal allowable information should be provided here, such as details of the distribution over various factors in the training datasets.
- **Quantitative Analyses**
  - Unitary results
  - Intersectional results
- **Ethical Considerations**
- **Caveats and Recommendations**

Exercise: ML Model Hunt

Search the web to find the model card for a real-world model

Model Card

- **Model Details**. Basic information about the model.
  - Person or organization developing model
  - Model date
  - Model version
  - Model type
  - Information about training algorithms, parameters, fairness constraints or other applied approaches, and features
  - Paper or other resource for more information
  - Citation details
  - License
  - Where to send questions or comments about the model
- **Intended Use**. Use cases that were envisioned during development.
  - Primary intended uses
  - Primary intended users
  - Out-of-scope use cases
- **Factors**. Factors could include demographic or phenotypic groups, environmental conditions, technical attributes, or others listed in Section 4.3.
  - Relevant factors
  - Evaluation factors

Neural Network Toolkits

Pytorch

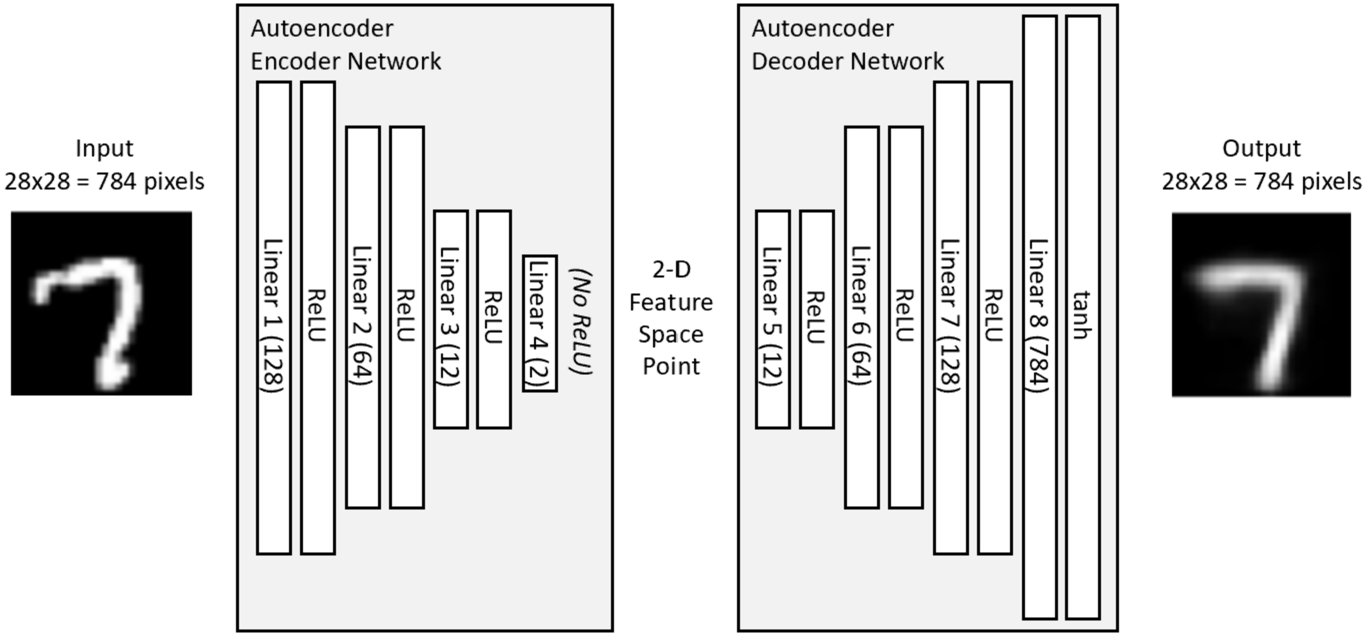
Network Network Toolkits

Pytorch in this course

- Already used behind the scenes in HW1
- HW7 transfer learning and exploration
- Mini-project if you want

Pytorch for HW1 Networks

Autoencoder



Source: https://github.com/L1aoXingyu/pytorch-beginner/blob/master/08-AutoEncoder/simple_autoencoder.py

```python
class autoencoder(nn.Module):
    def __init__(self, bottleneck=2):
        super().__init__()
        self.encoder = nn.Sequential(
            nn.Linear(28 * 28, 128),
            nn.ReLU(True),
            nn.Linear(128, 64),
            nn.ReLU(True),
            nn.Linear(64, 12),
            nn.ReLU(True),
            # nn.Linear(12, 2))
            nn.Linear(12, bottleneck))

        self.decoder = nn.Sequential(
            # nn.Linear(2, 12),
            nn.Linear(bottleneck, 12),
            nn.ReLU(True),
            nn.Linear(12, 64),
            nn.ReLU(True),
            nn.Linear(64, 128),
            nn.ReLU(True),
            nn.Linear(128, 28 * 28),
            nn.Tanh())

    def forward(self, x):
        x = self.encoder(x)
        x = self.decoder(x)
        return x
```

Pytorch for HW1 Networks

Autoencoder

```
for epoch in range(num_epochs):
    for data in dataloader:
        img, _ = data
        img = img.view(img.size(0), -1)
        if torch.cuda.is_available():
            img = img.cuda()
        # 
===================forward=====================
        output = model(img)
        loss = criterion(output, img)
        # 
===================backward====================
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```

Source: https://github.com/L1aoXingyu/pytorch-beginner/blob/master/08-AutoEncoder/simple_autoencoder.py

```
class autoencoder(nn.Module):
    def __init__(self, bottleneck=2):
        super().__init__()
        self.encoder = nn.Sequential(
            nn.Linear(28 * 28, 128),
            nn.ReLU(True),
            nn.Linear(128, 64),
            nn.ReLU(True),
            nn.Linear(64, 12),
            nn.ReLU(True),
            # nn.Linear(12, 2))
            nn.Linear(12, bottleneck))

        self.decoder = nn.Sequential(
            # nn.Linear(2, 12),
            nn.Linear(bottleneck, 12),
            nn.ReLU(True),
            nn.Linear(12, 64),
            nn.ReLU(True),
            nn.Linear(64, 128),
            nn.ReLU(True),
            nn.Linear(128, 28 * 28),
            nn.Tanh())

    def forward(self, x):
        x = self.encoder(x)
        x = self.decoder(x)
        return x
```

Pytorch for HW1 Networks

Autoencoder

```
for epoch in range(num_epochs):
    for data in dataloader:
        img, _ = data
        img = img.view(img.size(0),
        if torch.cuda.is_available()
            img = img.cuda()
        #
===================forward=========
        output = model(img)
        loss = criterion(output, im
        #
===================backward========
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```

```
class MiniImageCSVDataset:
    def __init__(self, csv_filename, has_labels=True, header=0):
        df = pd.read_csv(csv_filename, header=header)
        data = df.values
        if has_labels:
            self.data = data[:, 1:]
            self.labels = data[:, 0]
        else:
            self.data = data
            self.labels = None

        self.data = self.data.astype('float32')
        self.data = self.data / 255

    def __len__(self):
        return len(self.data)

    def __getitem__(self, idx):
        if self.labels is not None:
            return self.data[idx], self.labels[idx]
        else:
            return self.data[idx]
```

Network Network Toolkits

More PyTorch to come in recitation (and HW7)!