Introduction to CNNs and RNNs with PyTorch

Presented by: Adam Balint

Email: balint@uoguelph.ca

Working with more complex data

- Images
- Videos
- Sound
- Time Series
- Text

Task 1:

- Image classification

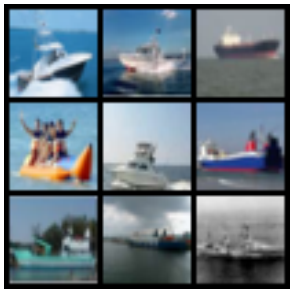
Image Source (https://www.designboom.com/cars/water-car-python/)

Dataset

- CIFAR10
  - 10 classes
  - 32 x 32 pixel image size
  - 60.000 examples
    - 50.000 Training
    - 10.000 Testing

0 - Airplane

1 - Automobile



8 - Ship

9 - Truck

Intro to Convolutional Neural Networks (CNNs)

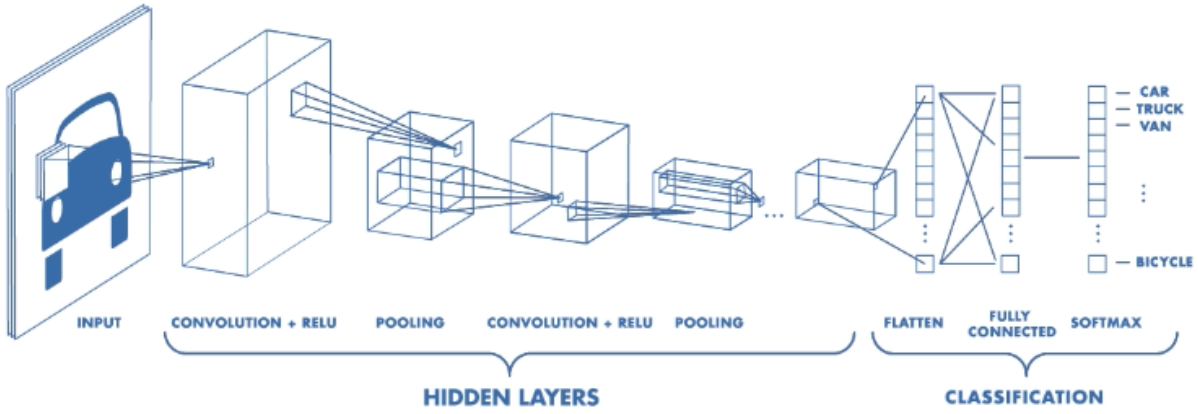


Image Source (https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html)

CNN Architecture Component - Convolutional Layer

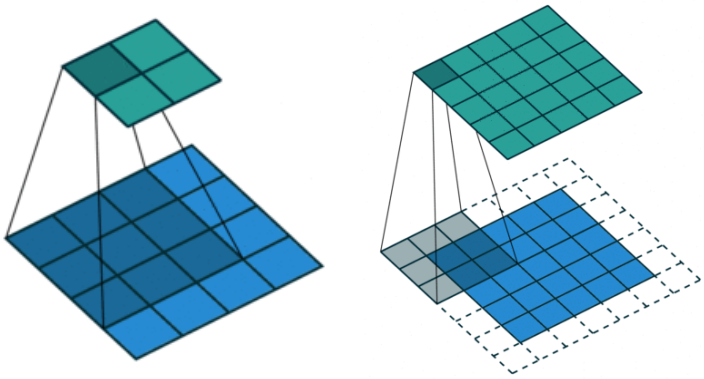


Image 1	Image 2
Kernel size: 3	Kernel size: 3
Stride size: 1	Stride size: 1
Padding size: 0	Padding size: 1

More visualizations can be seen here (https://github.com/vdumoulin/conv_arithmetic)

In [ ]:
```
nn.Conv2d(cin, cout, kernel_size=3, stride=1, padding=0)

nn.Conv2d(cin, cout, kernel_size=3, stride=1, padding=1)
```

CNN Architecture Components - Pooling Layer

Image Source (https://www.quora.com/What-is-max-pooling-in-convolutional-neural-networks)

In [ ]:
```
nn.MaxPool2d(kernel_size=2, stride=2)
```

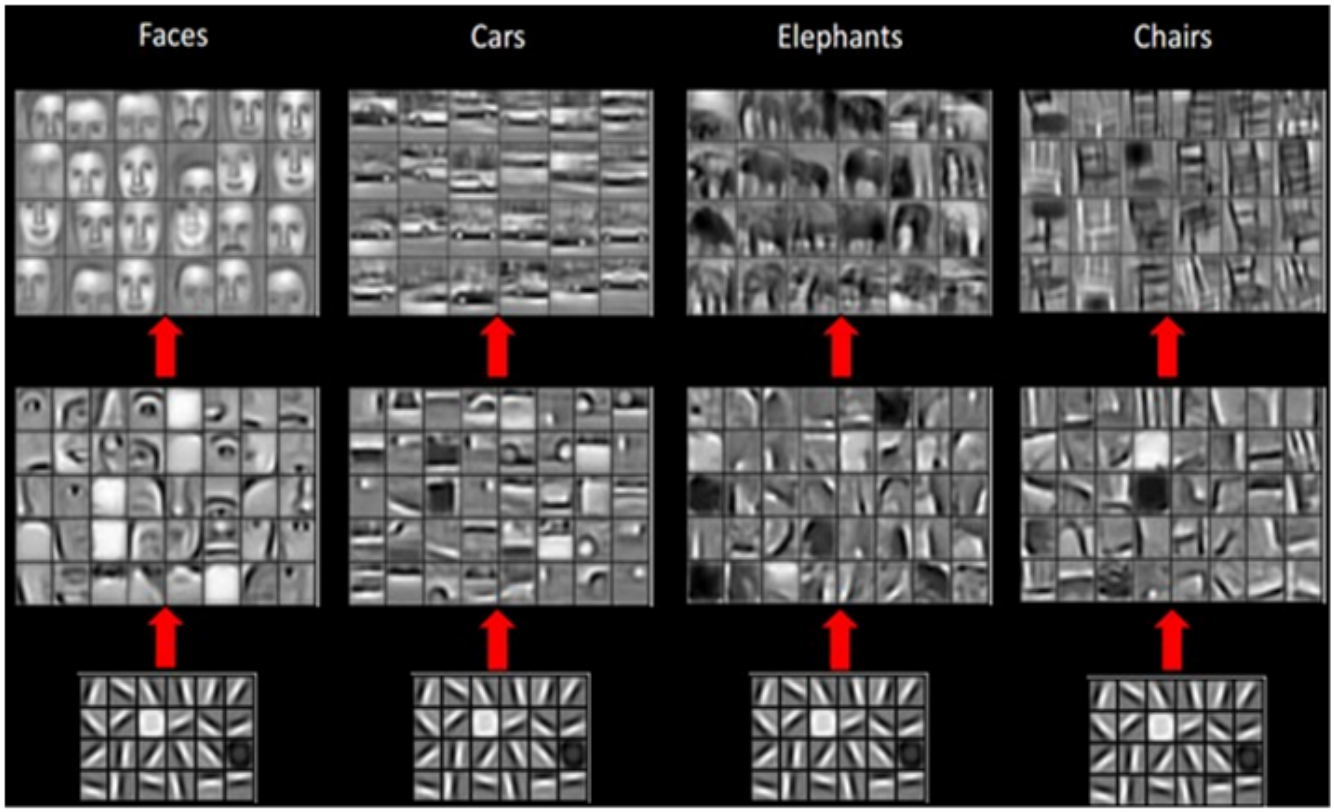


Image Source (https://i.stack.imgur.com/Hl2H6.png)

Implementing a CNN

In [ ]:
```
def conv_block(cin, cout, batch_norm=True, activation=nn.ReLU):
    if batch_norm:
        return nn.Sequential(
                    nn.Conv2d(cin, cout, kernel_size=3, stride=1, padding=0),
                    nn.BatchNorm2d(cout),
                    activation())
    return nn.Sequential(
                nn.Conv2d(cin, cout, kernel_size=3, stride=1, padding=0),
                activation())
```

In [ ]:
```
class ConvNet(nn.Module):
    def __init__(self, inp_size, out_size):
        super(ConvNet, self).__init__()
        self.extractor = nn.Sequential(
                conv_block(inp_size, 16),
                nn.MaxPool2d(kernel_size=2, stride=2),
                conv_block(16, 32),
                nn.MaxPool2d(kernel_size=2, stride=2),
                )
        self.classifier = nn.Sequential(
                nn.Linear(32*6*6, 100),
                nn.ReLU(),
                nn.Linear(100, out_size),
                nn.Sigmoid())

    def forward(self, inp):
        out = self.extractor(inp)
        out = out.view(out.size(0), -1)
        out = self.classifier(out)
        return out
```

Task 2:

- Sentiment Analysis

I think it's one of the greatest movies which are ever made, and I've seen many... The book is better, but it's still a very good movie!

Positive

Were I not with friends, and so cheap, I would have walked out. It failed miserably as satire and didn't even have the redemption of camp.

Negative

Dataset

- IMDB Review
- 50.000 examples
  - 25.000 training
  - 25.000 Testing
- Labels: Positive (1) and Negative (0)

Intro to Recurrent Neural Networks (RNNs)

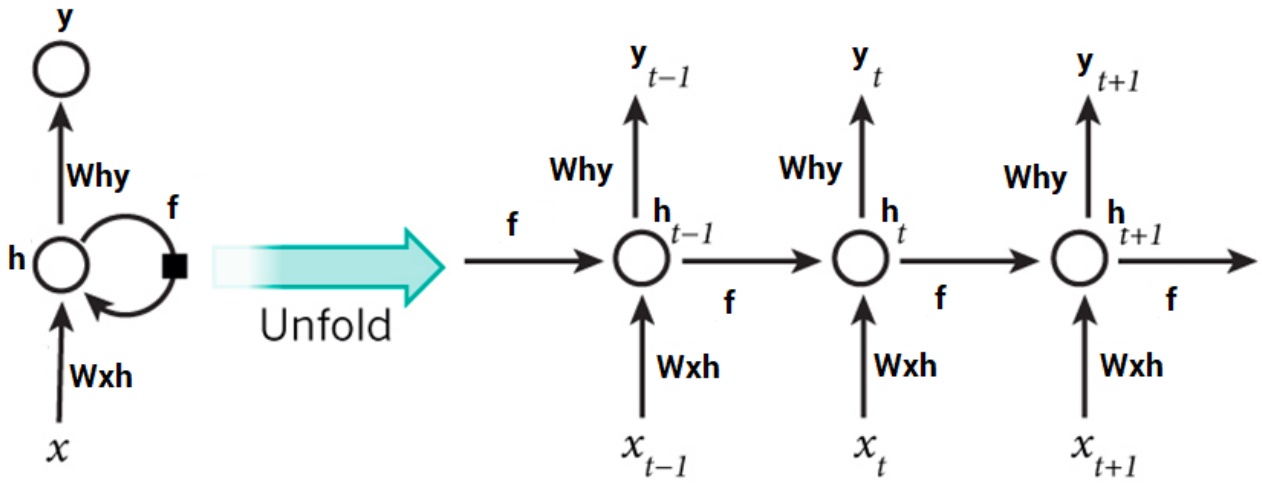


Image Source (https://www.analyticsvidhya.com/blog/2017/12/introduction-to-recurrent-neural-networks/)

RNN Types

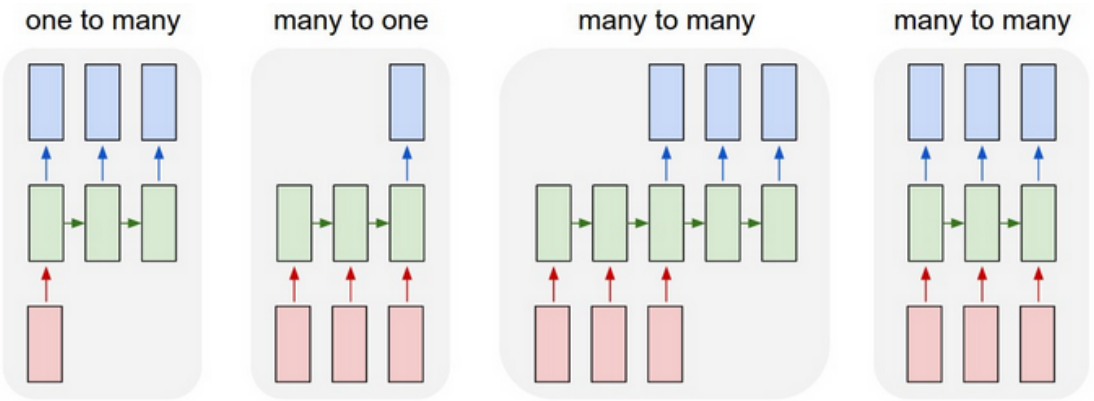
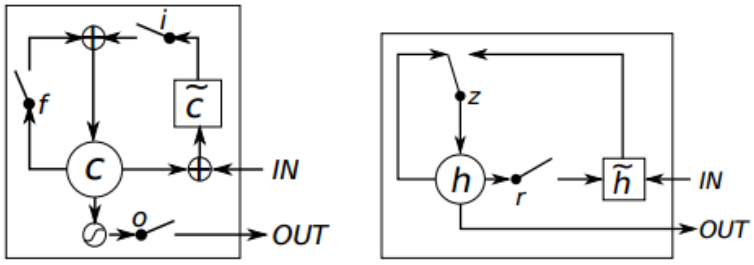


Image Source (http://karpathy.github.io/2015/05/21/rnn-effectiveness/)

RNN Architecture Components - Memory Units



(a) Long Short-Term Memory (b) Gated Recurrent Unit

Figure 1: Illustration of (a) LSTM and (b) gated recurrent units. (a) i, f and o are the input, forget and output gates, respectively. c and $\tilde{c}$ denote the memory cell and the new memory cell content. (b) r and z are the reset and update gates, and h and $\tilde{h}$ are the activation and the candidate activation.

Image Source (https://deeplearning4j.org/lstm.html)

In [ ]:
```
nn.LSTMCell(inp_dim, hid_dim)
nn.LSTM(inp_dim, hid_dim)

nn.GRUCell(inp_dim, hid_dim)
nn.GRU(inp_dim, hid_dim)
```

Implementing a RNN

In [ ]:
```
class LSTM(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(LSTM, self).__init__()

        self.embedding = nn.Embedding(input_size, 500)
        self.lstm = nn.LSTM(500, hidden_size, num_layers=1, bidirectional=True)
        self.fc = nn.Linear(hidden_size*2, output_size)

    def forward(self, inp):
        embedded = self.embedding(inp)
        output, (hidden, cell) = self.lstm(embedded)
        hidden = torch.cat((hidden[-2,:,:], hidden[-1,:,:]), dim=1)
        return self.fc(hidden.squeeze(0))
```

In [ ]:
```
class GRU(nn.Module):
    def __init__(self, inp_dim, hid_dim, out_dim):
        super(RNN, self).__init__()
        self.embedding = nn.Embedding(inp_dim, 100)
        self.rnn = nn.GRU(100, hid_dim, bidirectional=False)
        self.fc = nn.Linear(hid_dim, out_dim)

    def forward(self, inp):
        embedded = self.embedding(inp)
        output, hidden = self.rnn(embedded)
        return self.fc(hidden.squeeze(0))
```

Training the Networks

- Prepare the dataset
- Set up training components
- Create training loop
- Test network

Prepare the Dataset

In [ ]:
```
transform = transforms.ToTensor()
training_data = datasets.CIFAR10(dataset_location, download=True, transform=transf
orm)
testing_data = datasets.CIFAR10(dataset_location, train=False, download=True, tran
sform=transform)
training_data, validation_data = random_split(training_data, lengths=[len(training
_data)-1000, 1000])

train_loader = DataLoader(training_data, shuffle=True, batch_size=batch_size)
val_loader = DataLoader(validation_data, batch_size=batch_size)
test_loader = DataLoader(testing_data, batch_size=batch_size)
```

In [ ]:
```
train = IMDB(os.environ['HOME'] + "/shared/dataset/train_dl.pkl")
val = IMDB(os.environ['HOME'] + "/shared/dataset/val_dl.pkl")
test = IMDB(os.environ['HOME'] + "/shared/dataset/test_dl.pkl")

train_iter, valid_iter, test_iter = data.BucketIterator.splits((train, val, test),
        batch_size=BATCH_SIZE,
        sort_key=lambda x: len(x.text),
        repeat=False)
```

Set up Training Components

In [ ]:
```
model = ConvNet(3, 4).cuda()
optim = torch.optim.SGD(model.parameters(), lr=0.01, momentum=0)
criterion = nn.BCELoss()
```

In [ ]:
```
model = RNN(INPUT_DIM, HIDDEN_DIM, OUTPUT_DIM).cuda()
optim = torch.optim.Adam(model.parameters(), lr=0.05)
criterion = nn.BCEWithLogitsLoss()
```

Creating the Training Loop

In [ ]:
```
def train_epoch(model, iterator, optimizer, criterion):
    epoch_loss = 0
    epoch_acc = 0

    model.train()
    for data in iterator:
        optimizer.zero_grad()
        x = data[0].cuda()
        y = torch.zeros(x.size(0), len(class_subset)).float()
        y = y.scatter_(1, data[1].view(x.size(0), 1), 1.0).cuda()
        predictions = model(x)

        loss = criterion(predictions, y)
        acc = calculate_accuracy(predictions, y)

        loss.backward()
        optimizer.step()

        epoch_loss += loss.item()
        epoch_acc += acc.item()

    return epoch_loss / len(iterator), epoch_acc / len(iterator)
```

Creating the Training Loop

In [ ]:
```
def train_epoch(model, iterator, optimizer, criterion):
    epoch_loss = 0
    epoch_acc = 0

    model.train()

    for batch in iterator:
        optimizer.zero_grad()
        predictions = model(batch.text).squeeze(1)

        y = batch.label

        loss = criterion(predictions, y)
        acc = calculate_accuracy(predictions, y)

        loss.backward()
        optimizer.step()

        epoch_loss += loss.item()
        epoch_acc += acc.item()

    return epoch_loss / len(iterator), epoch_acc / len(iterator)
```

Training the Network

In [ ]:
```
for epoch in range(num_epochs):
    train_loss, train_acc = train_epoch(model, train_iter, optim, criterion)
    valid_loss, valid_acc = evaluate_epoch(model, valid_iter, optim, criterion)
```

Testing the Network

In [ ]:
```
test_loss, test_acc = evaluate_epoch(model, test_iter, optim, criterion)
```

Experiment Time and Questions

- Open the Intro to Convolutional Networks or Intro to Recurrent Networks notebook
  - Scroll to the Change Hyperparameters section of the notebook
  - Change the hyperparameters to try to improve the test time accuracy of the network

Scores to Beat

- Convolutional Networks: ~75%
- Recurrent Networks: ~70%

Feel free to ask questions

Further Applications

- 99.99% Surprised

We were barely able to catch the breeze at the beach , and it felt as if someone stepped out of my mind . She was in love with him for the first time in months , so she had no intention of escaping . The sun had risen from the ocean , making her feel more alive than normal . She 's beautiful , but the truth is that I do n't know what to do . The sun was just starting to fade away , leaving people scattered around the Atlantic Ocean . I d seen the men in his life , who guided me at the beach once more .
Samim has made an awesome blog post with lots of results here.

Emotion Detection (https://github.com/co60ca/EmotionNet) Style Transfer (https://github.com/AdamBalint/Picassos-Iris) Text to Image (https://github.com/aelnouby/Text-to-Image-Synthesis) Image to Text (https://github.com/ryankiros/neural-storyteller)