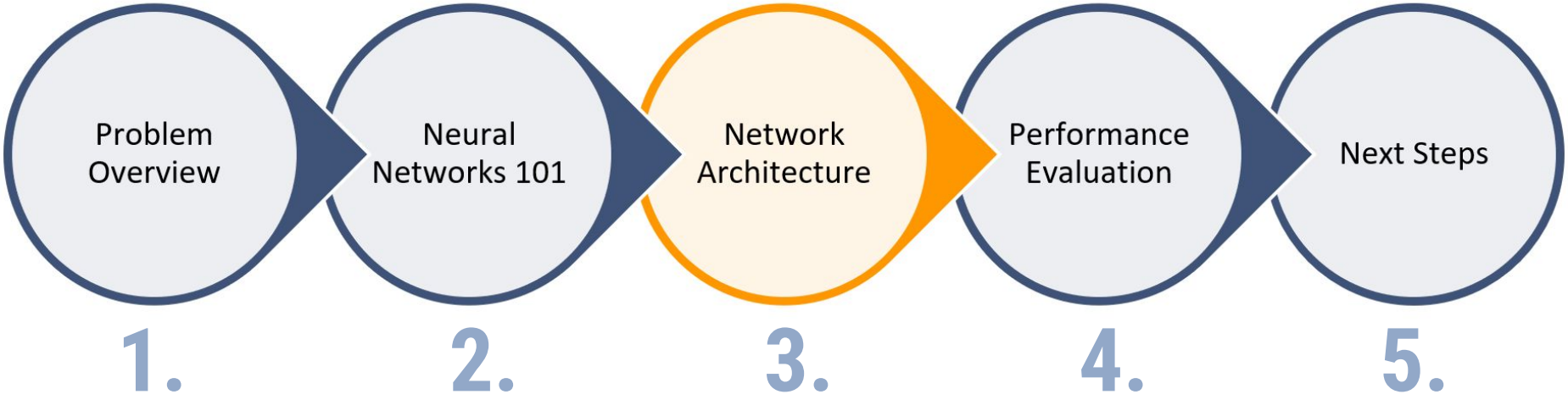
Lego Figurine Classification

w/ Convolutional Neural Networks

Week 3 Launch Final Project: Cooper, Key, Tohti 1

Agenda


Problem Overview
Neural Networks 101
Network Architecture
Performance Evaluation
Next Steps
1.
2.
3.
4.
5.

1.

Problem Definition

What are we trying to accomplish?

Problem: Label Lego Minifigures!

Harry Potter

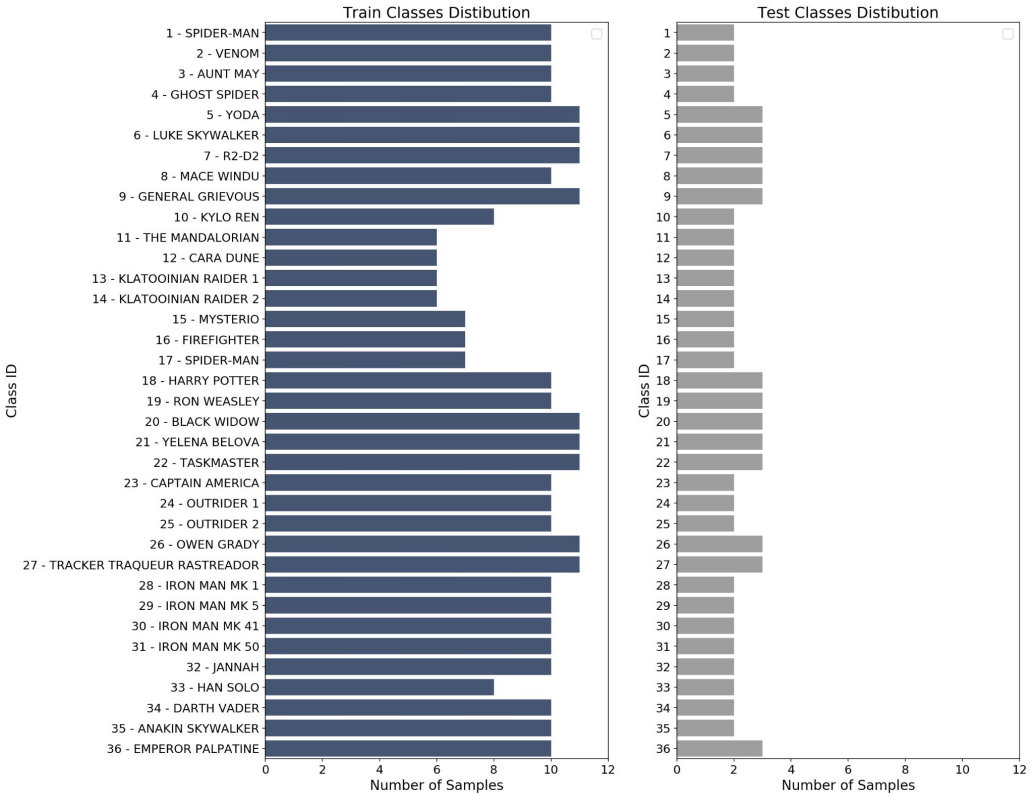
Spiderman

Harry Potter

Spiderman

Harry Potter

Data distribution



425

pictures total

85

test pictures

340

train pictures

2.

Convolutional Neural Networks 101

Understanding the process

Why Choose a Convolutional Neural Network?

- Best at **recognizing** different features about **image data**.
- Picks up on **edges, curves, outlines** and colors of image by applying image “convolutions” to the data.
- Is able to **classify images** into distinct categories

CNN Image preprocessing steps

1. Original Input Image

2. Vectorize Image

[0,1,0,1]

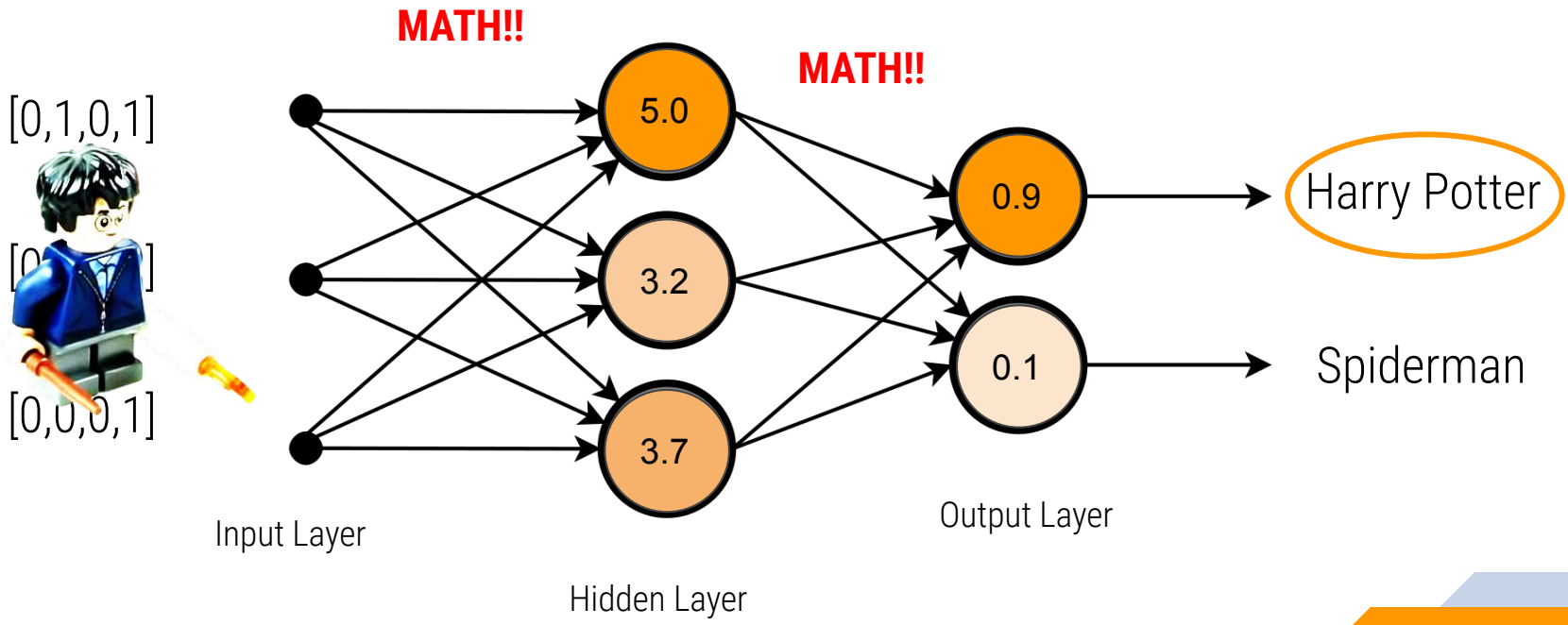
[0,1,1,1]

[0,0,0,1]

3. Resize Image Vecs

4. Normalize Vecs

How does a Neural Network actually Work?

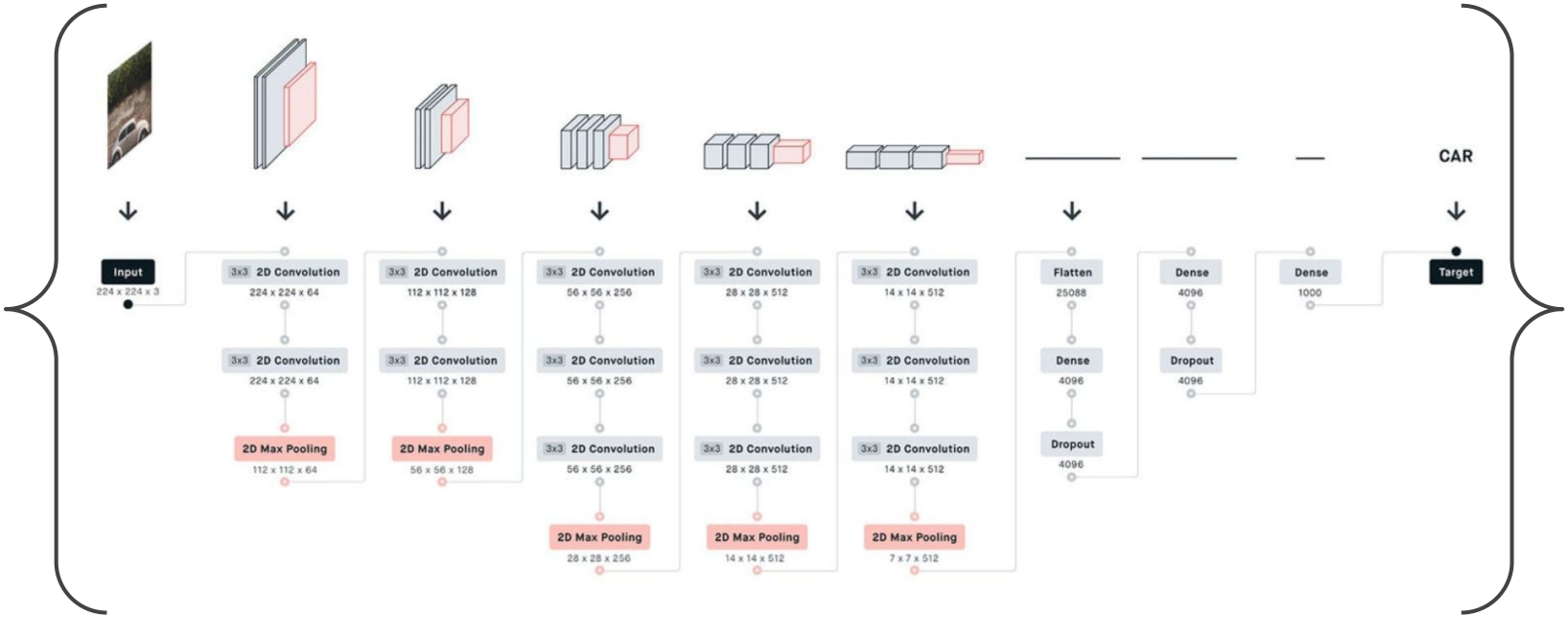


3.

Network Architecture

How did we design our model?

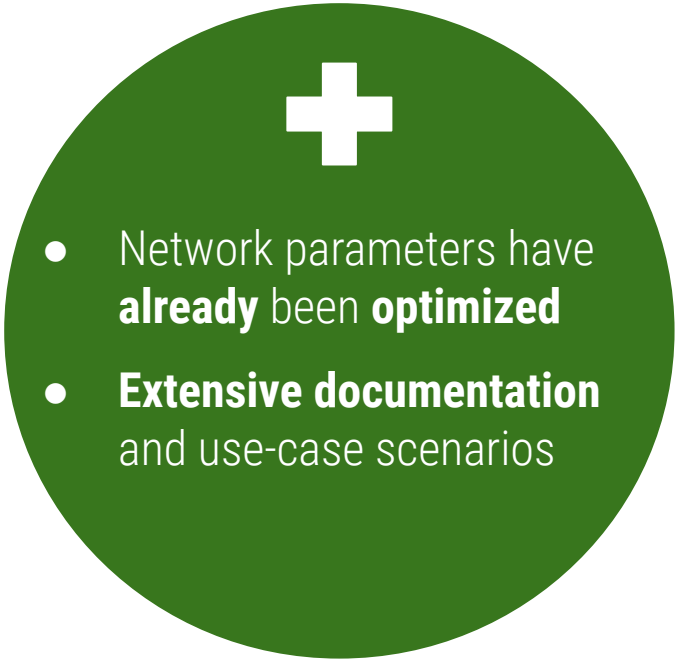
What is Transfer Learning?



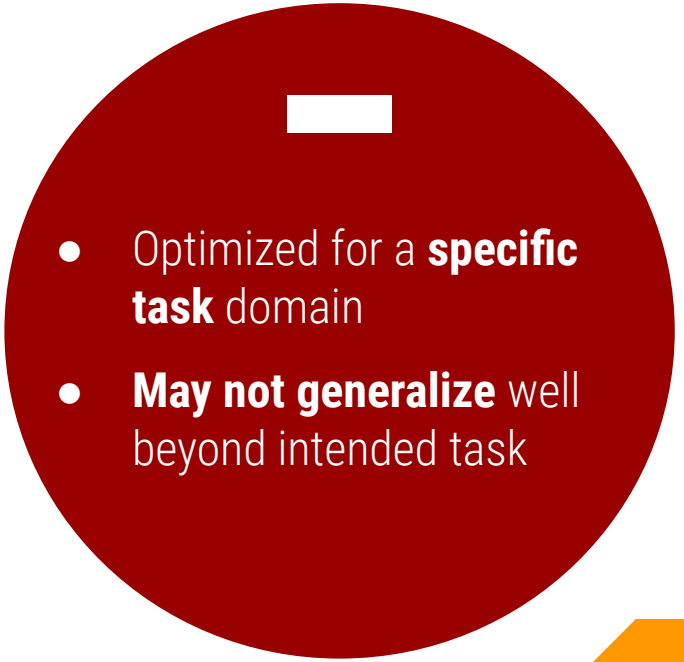
Resnet-18

Utilizing **pre-existing neural network architectures** to accomplish a similar task.

Pros and Cons of Transfer Learning



- Network parameters have **already** been **optimized**
- **Extensive documentation** and use-case scenarios



- Optimized for a **specific task** domain
- **May not generalize** well beyond intended task

4.

Performance Evaluation

How well did we perform?

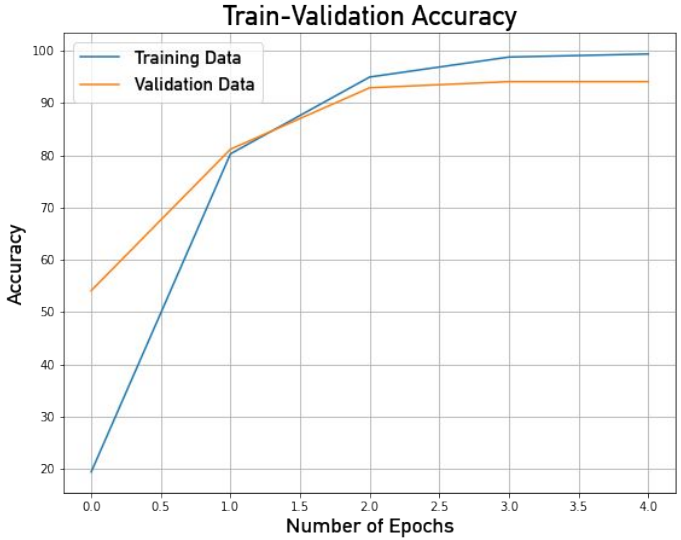
Choosing Performance Metrics

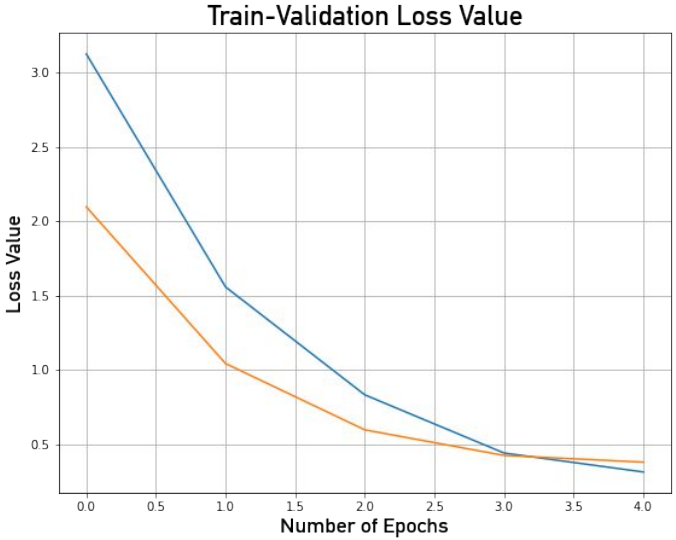
36 characters among **425** images, roughly evenly distributed by character. This means that we care equally about all categories and simply want to maximize the number of correct classifications of images to labels.

Thus, our primary metric will be **validation accuracy** which is simply the ratio of correctly classified images to total number of images.

$$Accuracy = \frac{Correct\ Predictions}{Total\ Predictions}$$

Resnet18 model performance





94%

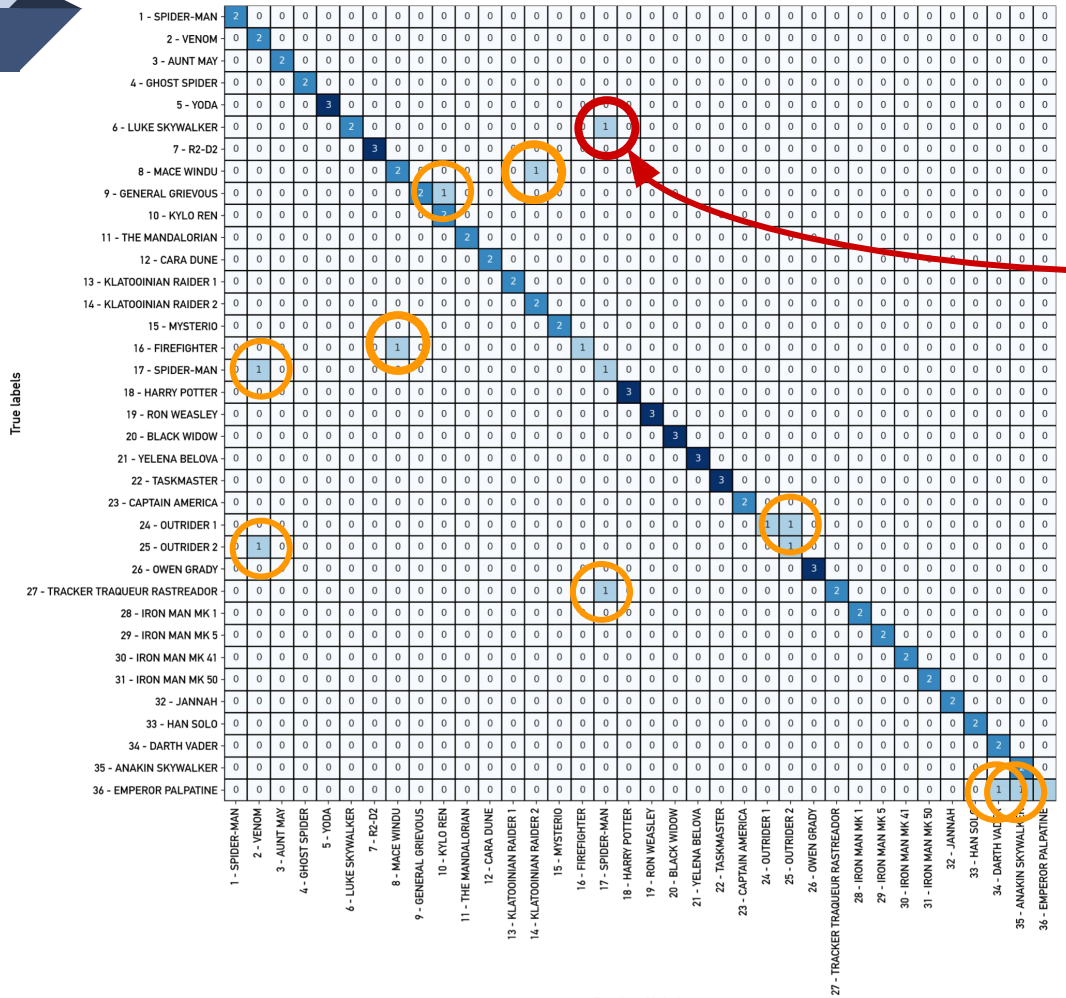
Validation Accuracy

99%

Training Accuracy

*Epoch = one pass of all of the data through the network

Confusion Matrix



Luke Skywalker Mistaken as Spiderman

Resnet18 model performance

Actual: R2-D2
Predicted: R2-D2

Actual: BLACK WIDOW
Predicted: BLACK WIDOW

Actual: RON WEASLEY
Predicted: RON WEASLEY

Actual: IRON MAN MK 1
Predicted: IRON MAN MK 1

6%

Incorrectly labeled images

Actual: GENERAL GRIEVOUS
Predicted: GENERAL GRIEVOUS

Actual: EMPEROR PALPATINE
Predicted: DARTH VADER

Actual: RON WEASLEY
Predicted: RON WEASLEY

Actual: YELENA BELOVA
Predicted: YELENA BELOVA

5.

Next Steps

What else can we do moving forward?

Next Steps

1. Attempt **other CNN transfer learning models** beyond Resnet18
2. **Tune hyperparameters** (learning rate, momentum, etc.) with GridSearch in order to improve model performance
3. **Generate synthetic images** for model to train on to increase prediction accuracy and generalizability
4. Implementing our own **custom model**
5. **Set seed** for **reproducibility**

Our favorite (Mis)classifications

Actual: OWEN GRADY
Predicted: GENERAL GRIEVOUS

Actual: TASKMASTER
Predicted: RON WEASLEY

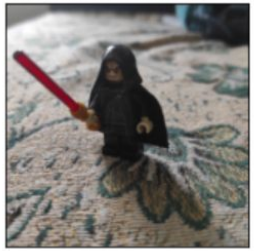
Actual: VENOM
Predicted: DARTH VADER

Actual: MYSTERIO
Predicted: HARRY POTTER

Actual: HARRY POTTER
Predicted: DARTH VADER

Actual: JANNAH
Predicted: IRON MAN MK 50

Actual: EMPEROR PALPATINE
Predicted: DARTH VADER



Actual: FIREFIGHTER
Predicted: RON WEASLEY

Questions?

We would be happy to provide answers :)

6.

Appendix

Etc, etc.

Building a Custom Model

Layers
Define the input, hidden and output layers of the CNN

Forward Pass
Define the order in which to implement the CNN layers and the functions to stabilize the signal through the net

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 64, kernel_size=(3, 3), padding=1)
        self.conv2 = nn.Conv2d(64, 64, kernel_size=(3, 3), padding=1)
        self.max_pool = nn.MaxPool2d(2, 2)
        self.global_pool = nn.AvgPool2d(7)
        self.fc1 = nn.Linear(64, 64)
        self.fc2 = nn.Linear(64, 10)

    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = F.relu(self.conv2(x))
        x = self.max_pool(x)

        x = F.relu(self.conv2(x))
        x = F.relu(self.conv2(x))
        x = self.max_pool(x)

        x = F.relu(self.conv2(x))
        x = F.relu(self.conv2(x))
        x = self.global_pool(x)

        x = x.view(-1, 64)

        x = F.relu(self.fc1(x))
        x = self.fc2(x)

        x = F.log_softmax(x)

        return x
model = Net()
```

Adding two convolution layers

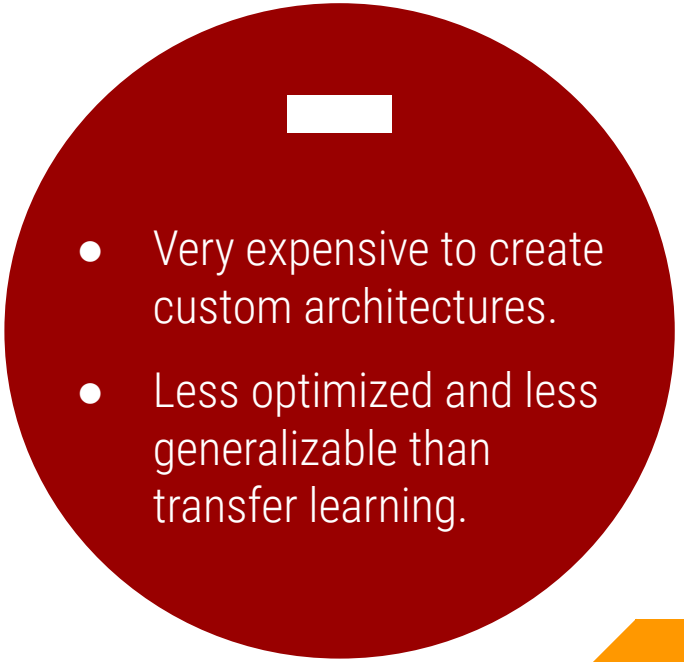
Activation Functions to turn the neurons ON or OFF based on prior input

Pool layer to reduce image dimensionality

Softmax to output classification probabilities

Implementation of Custom Model: Pros & Cons

- Greater understanding of the layers and how they interact.
- Very specific use cases for unusual data.



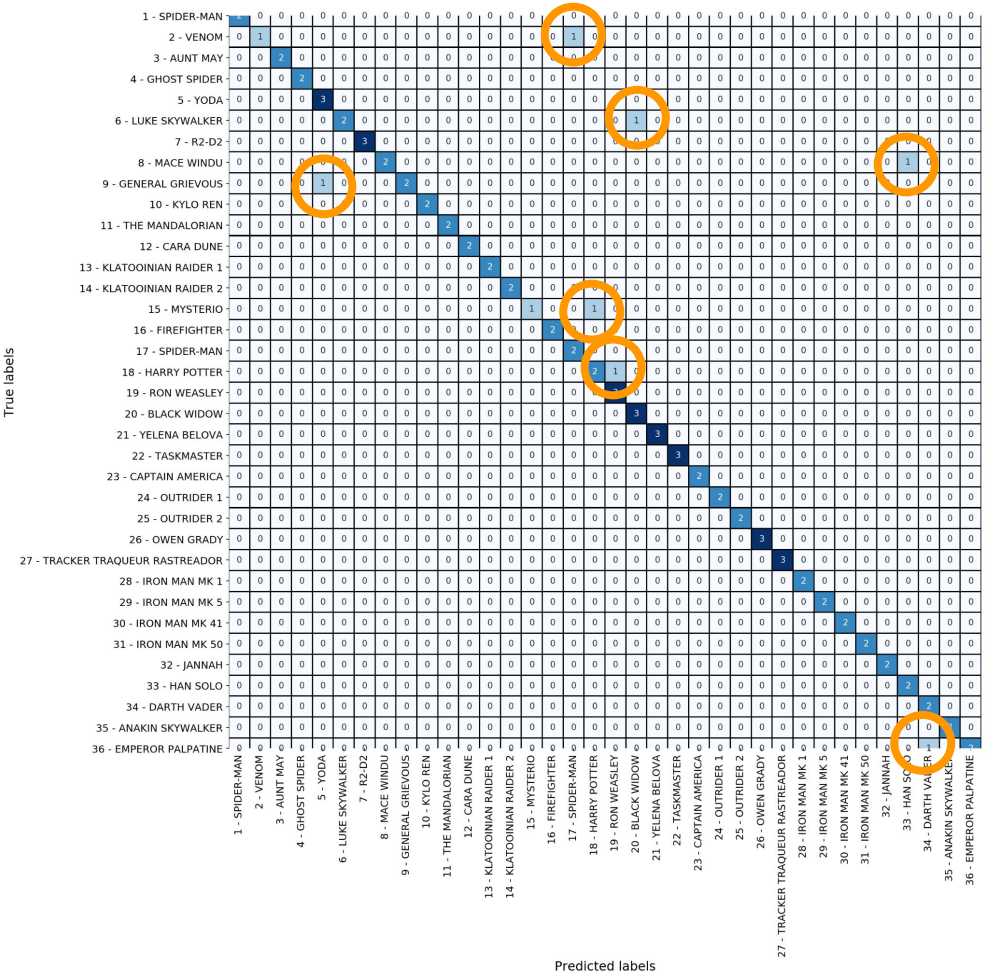
- Very expensive to create custom architectures.
- Less optimized and less generalizable than transfer learning.

Data Label Duplicates

Spiderman #1

Spiderman #2

Alt. Confusion Matrix

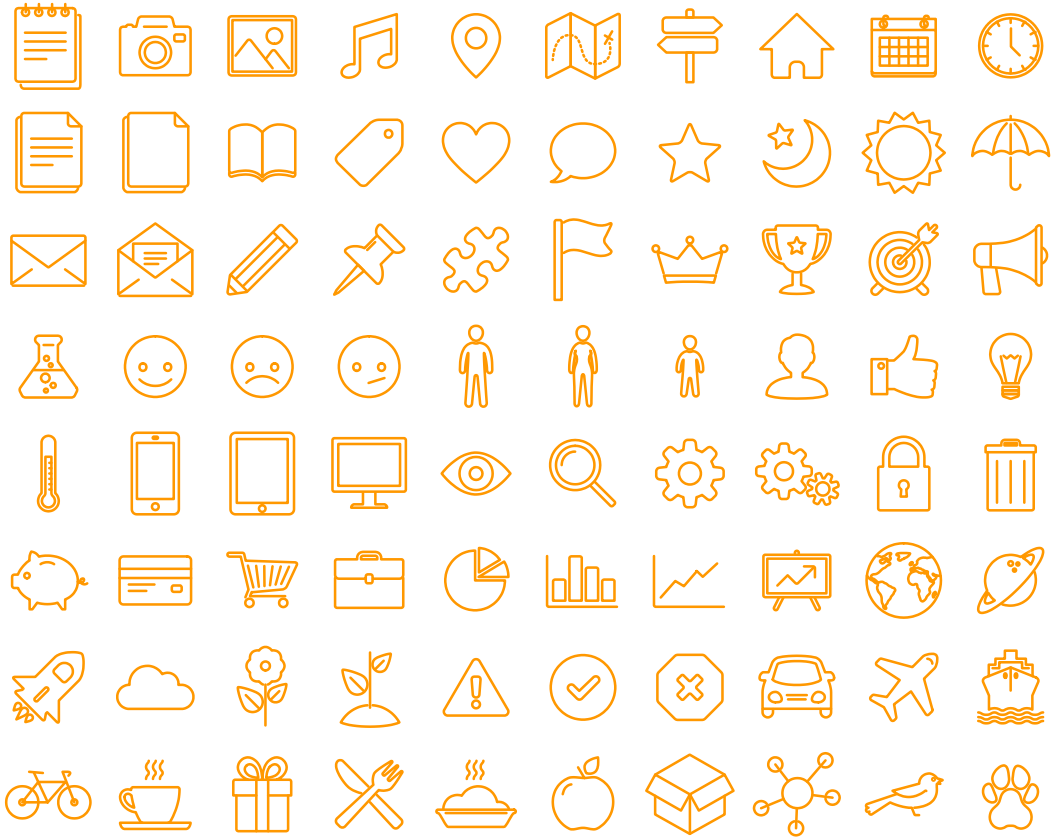


Running new confusion matrix runs existing network over new pictures.

Problem 2. Group Characters by Universe

Marvel Characters

Harry Potter Characters



SlidesCarnival icons are editable shapes.

This means that you can:

- Resize them without losing quality.
- Change line color, width and style.

Isn't that nice? :)

Examples: