Machine Learning Optimization of Photometric Redshift

Kate Sautel – Senior Talk 2

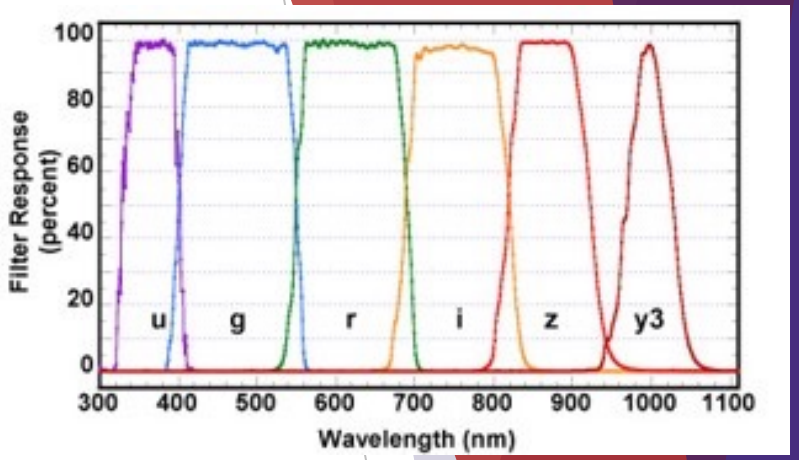
February 28, 2023

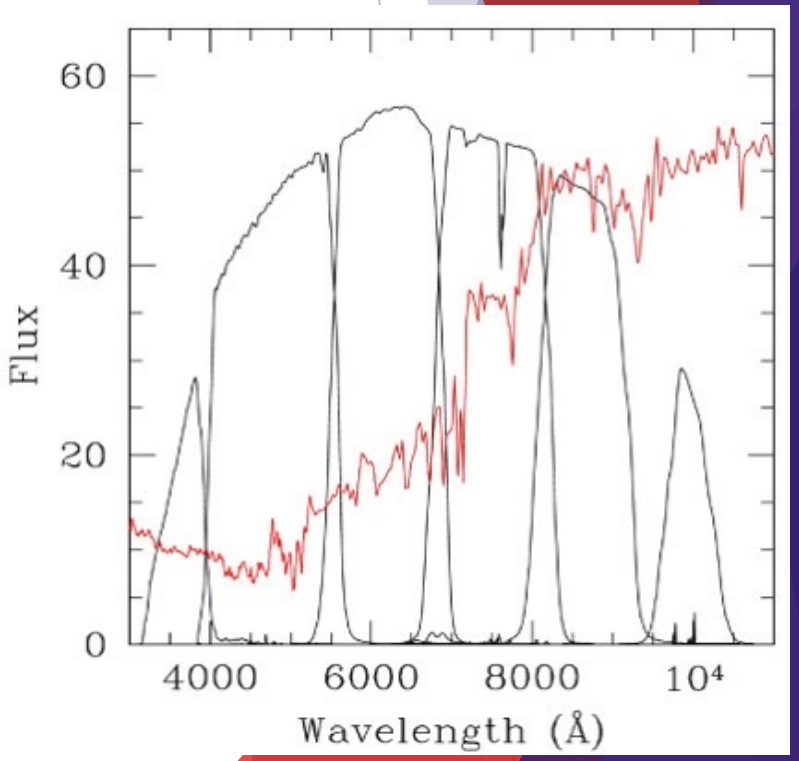
REVIEW

Photometric Redshift & Photometric Bands

- **"Spectroscopic"**: actual redshift
- **"Photometric"**: estimation of redshift from photometry
- **Photometry:** measure intensity (flux) of light emitted by object viewed through different filters
  - "band magnitudes"

Redshift changes → Spectrum shift (in f or λ) → Flux in photo bands changes





COs and NOs

- Categorize outliers into 3 categories
- Outlier:

$$O: \frac{\left|Z_{\text{phot}} - Z_{\text{spec}}\right|}{1 + Z_{\text{spec}}} > 0.15$$

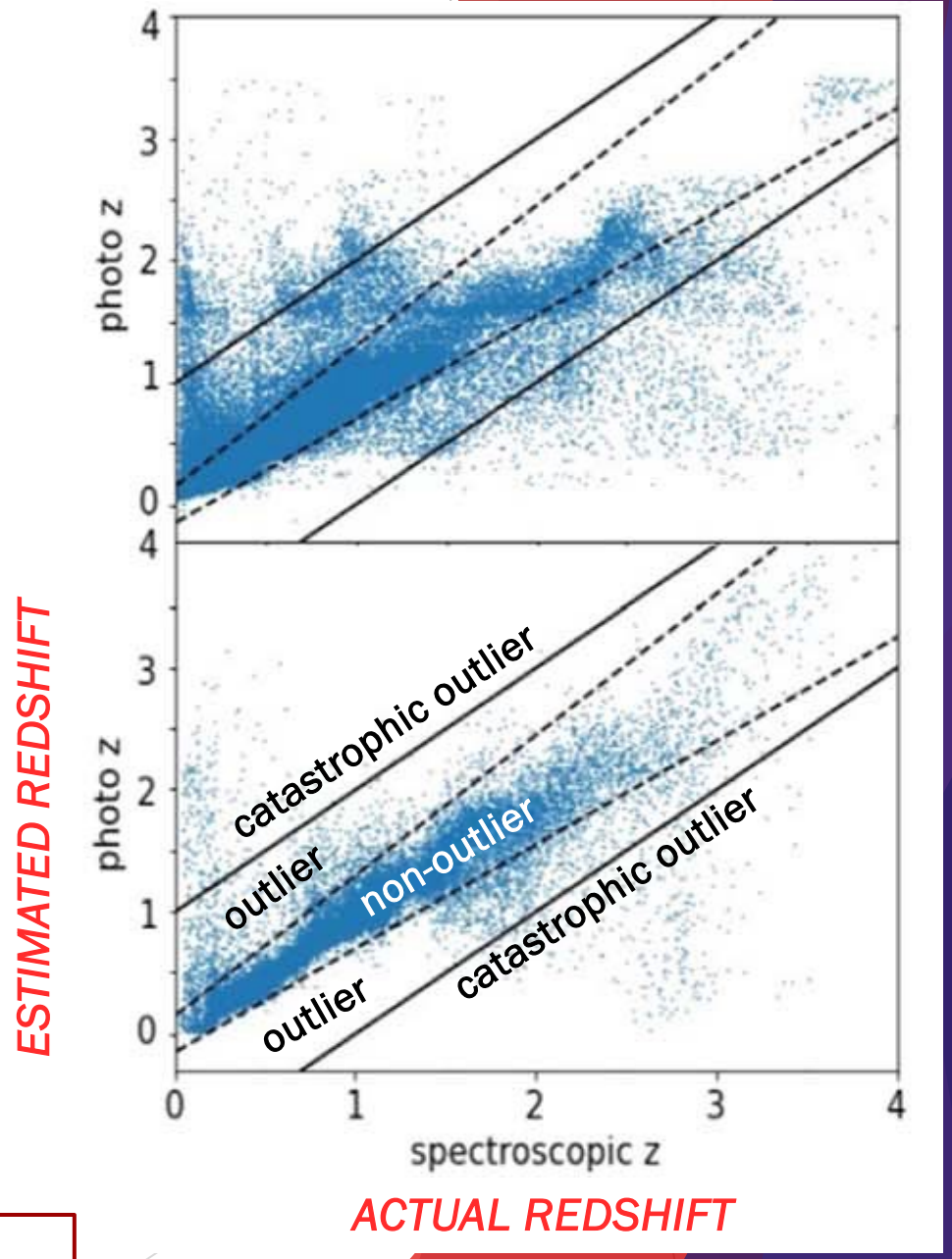
e.g. Hildebrandt, H. et al. 2010, *A&A*, 523, 832

- Catastrophic outlier:

$$O_c: \left|Z_{\text{phot}} - Z_{\text{spec}}\right| > 1.0$$

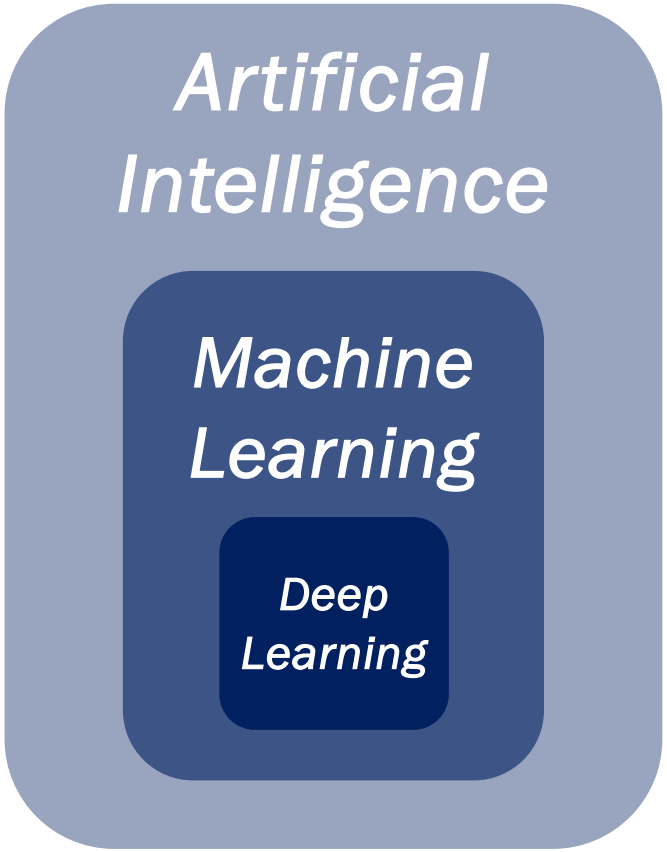
Bernstein, G. & Huterer, D. 2010, *MNRAS*, 401, 1399

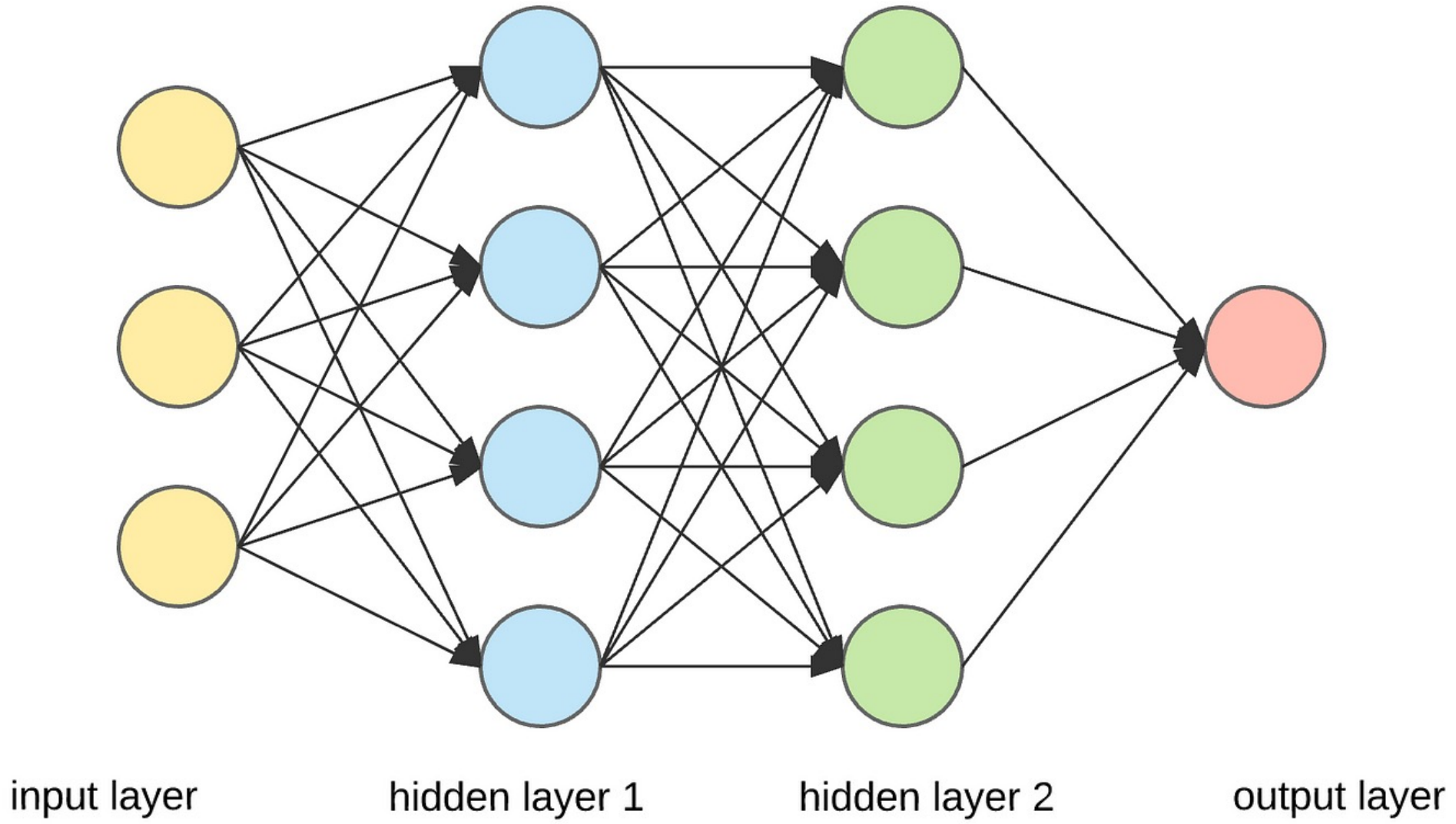
Z_{phot} = photometric redshift, Z_{spec} = spectroscopic redshift



Recap of Machine Learning

- Incidence of large-sky surveys
  - Spectro methods aren't feasible anymore
- Use large quantities of data to train machine to make simple correlations
  - Avoid *explicitly* coding for individual circumstances
- NNs are often backbone/structure
  - Most common




input layer
hidden layer 1
hidden layer 2
output layer

Input neurons: process/analyze/categorize input data and pass onto next layer

Hidden layer(s): further process input data or data from other hidden layers, weights attributed between neurons

Output layer: gives final “conclusion” of processing (0 or 1 or more complex)

Training Procedure

- **Training:** updating weights connecting neurons in hidden layers
  - Updated iteratively & automatically (learning rate)
- **Objective:** MINIMIZE loss function
  - Loss function: summed difference between targets (actual values) and predictions
    - Many different metrics
    - SGD, Adam, AdGrad

Parameters & Hyperparameters

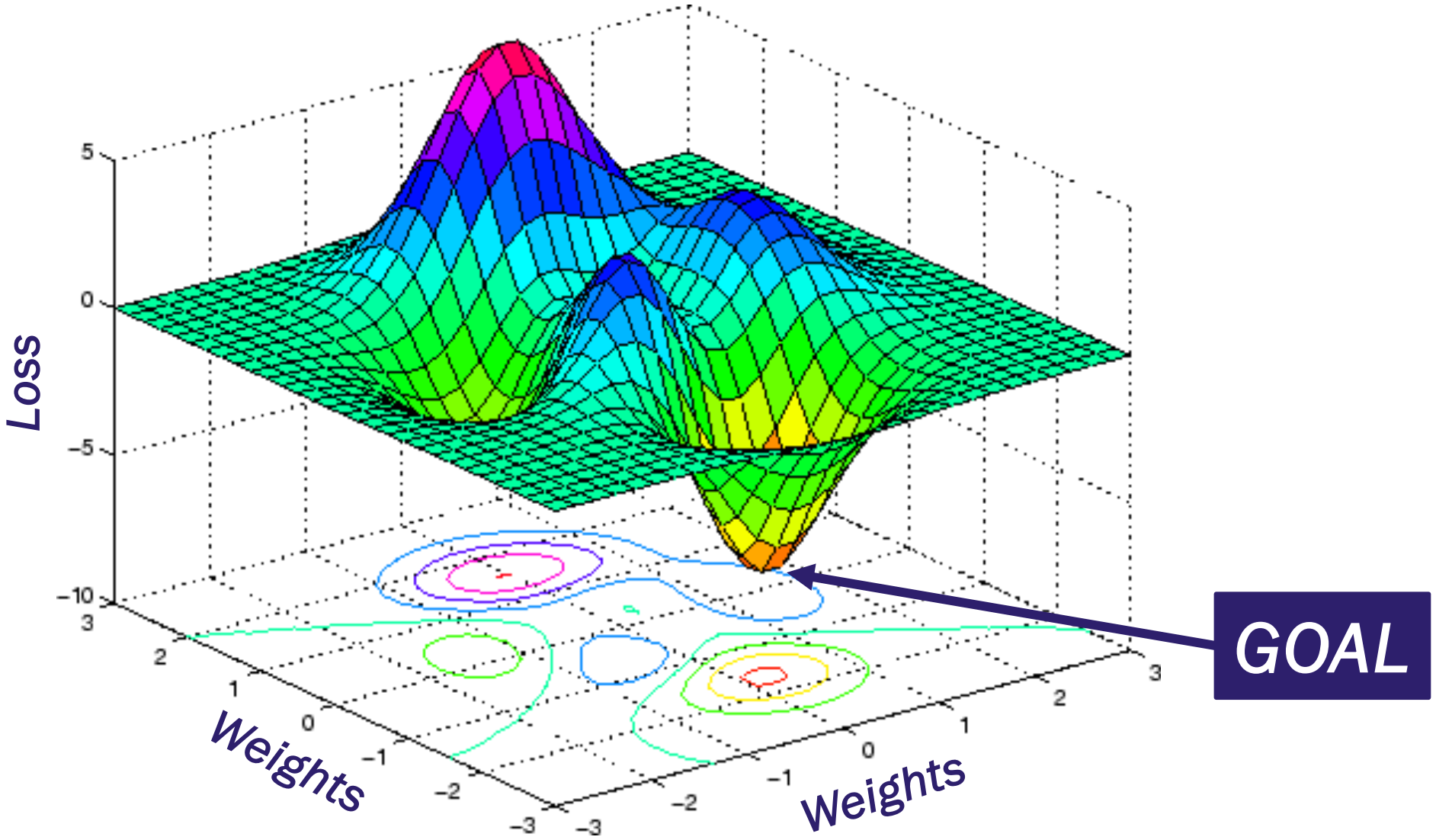
- **Parameters:** known inputs to input layers
  - Photo band magnitudes
- **Hyperparameters:** properties of the NN
  - Hidden layers
  - Neurons per layer
  - Loss metric
  - Momentum
  - Learning rate
  - Epochs
  - Batch size

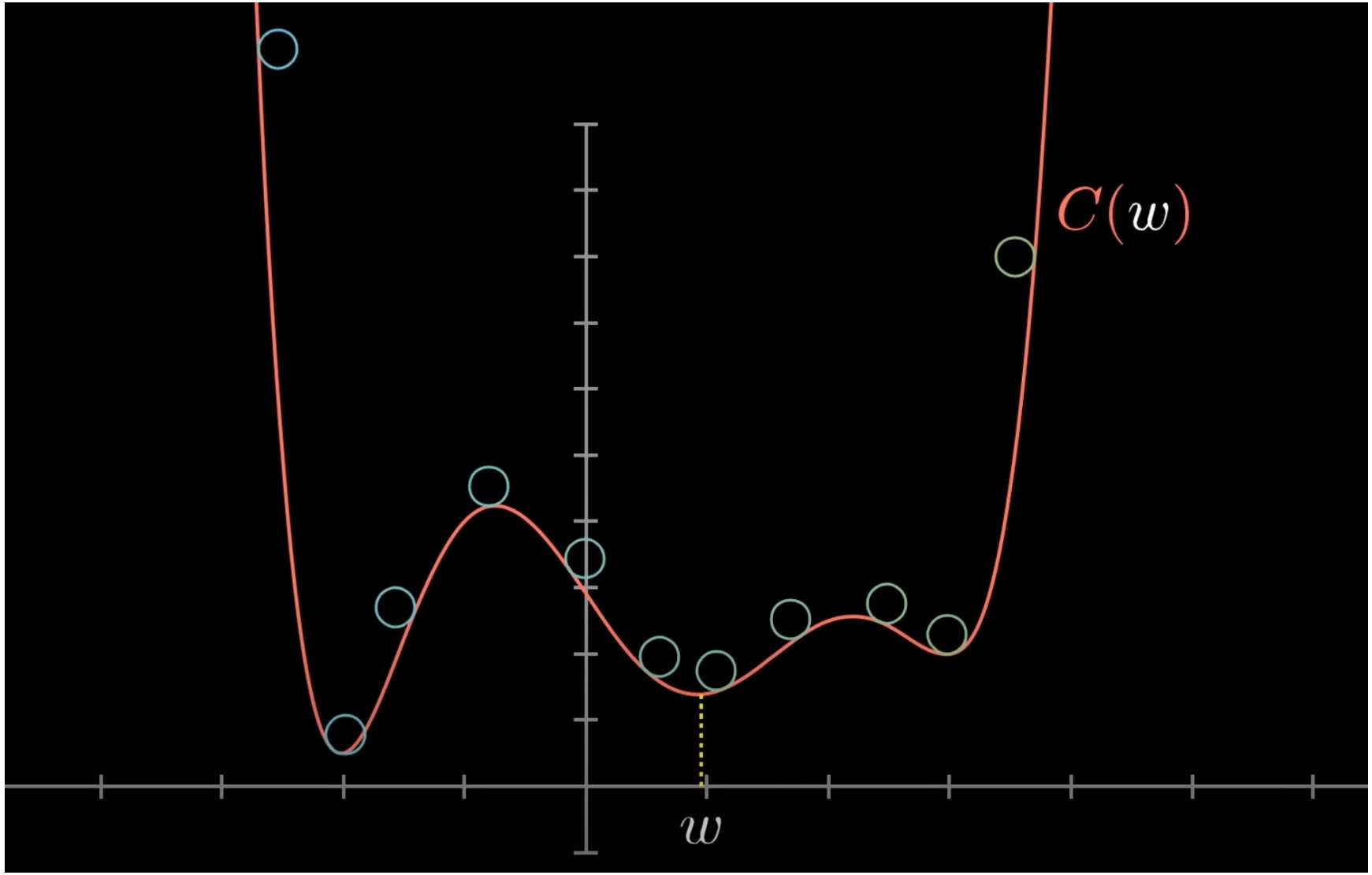
Training procedure parameters

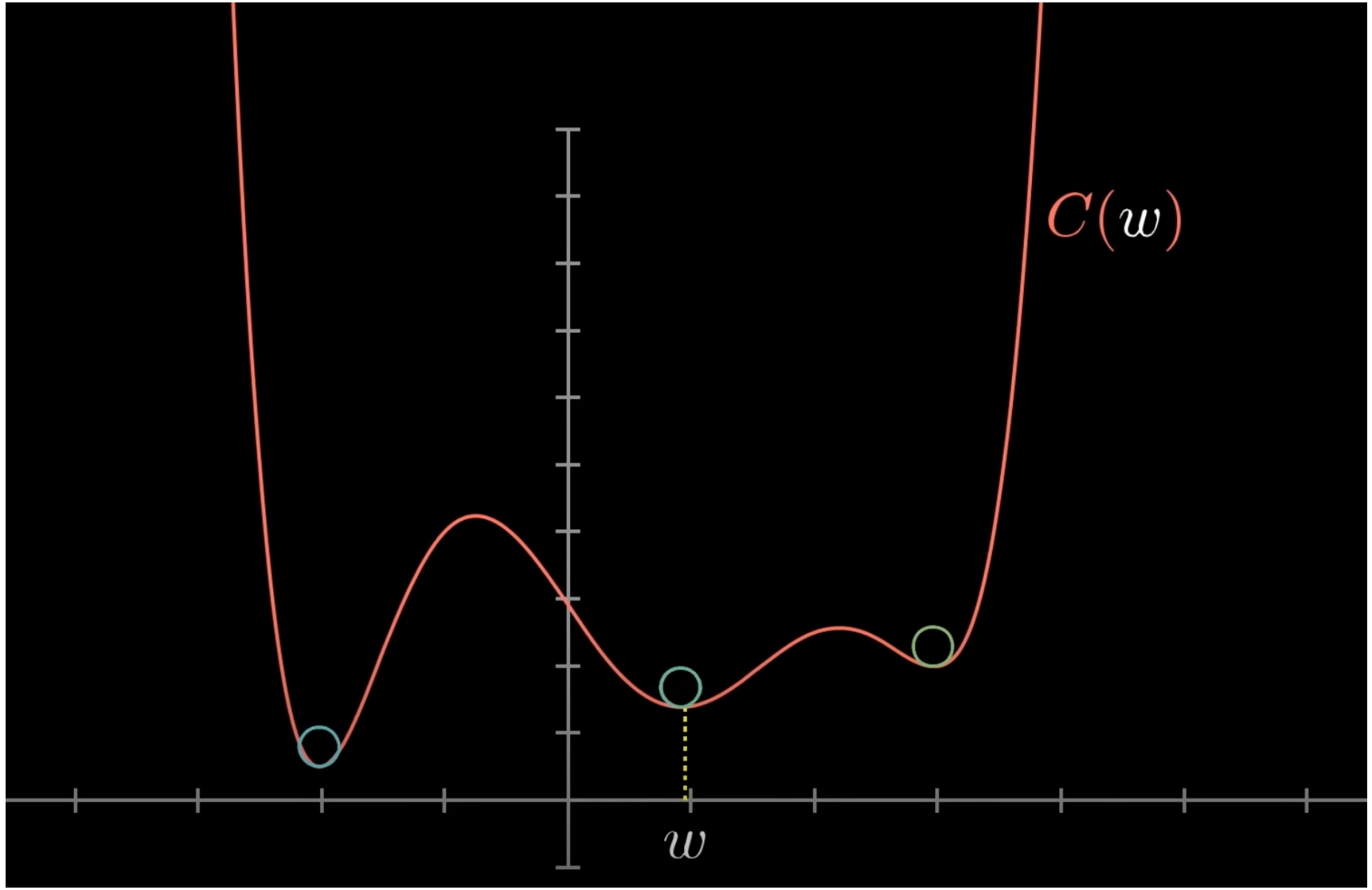
NEW MATERIAL

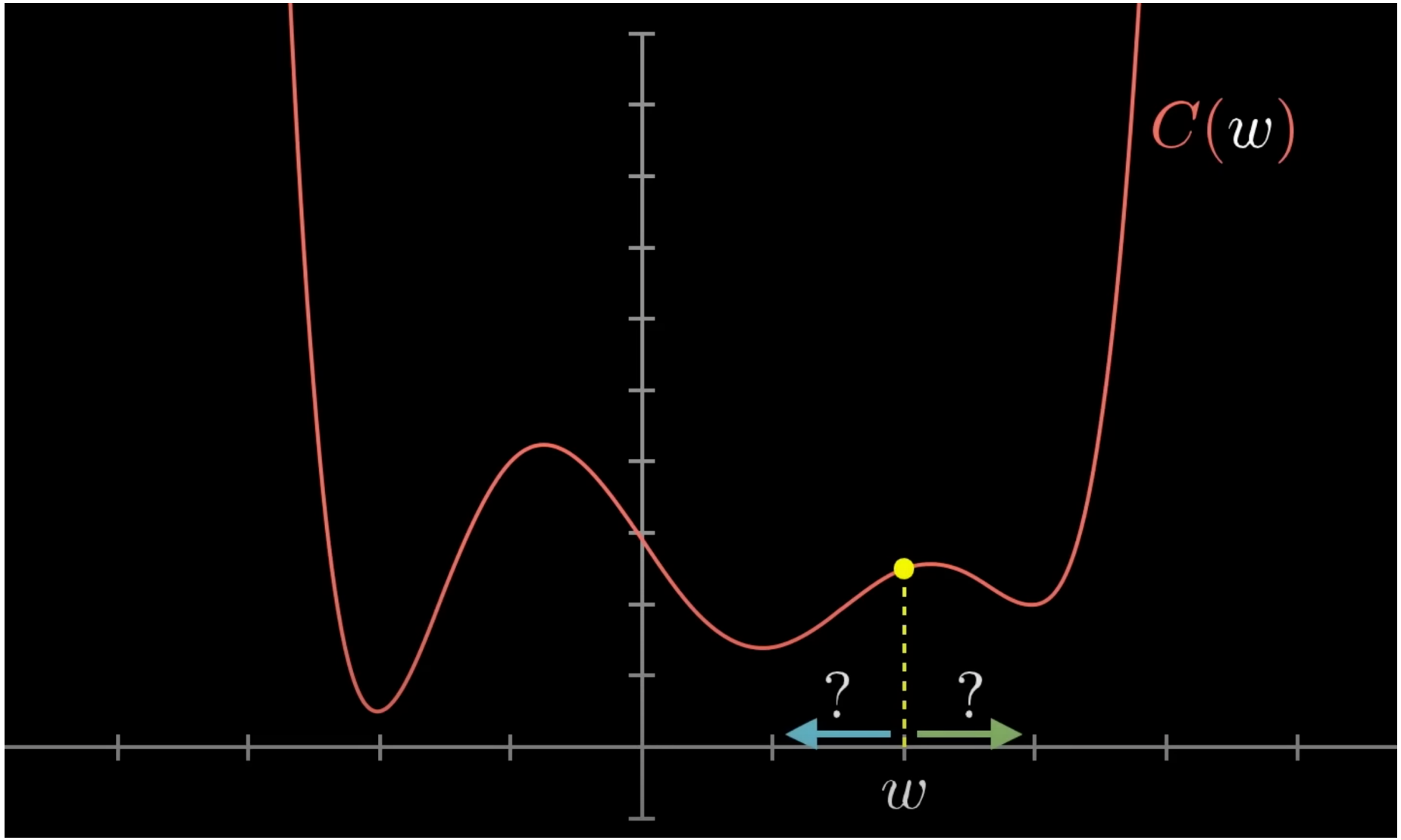
Backpropagation

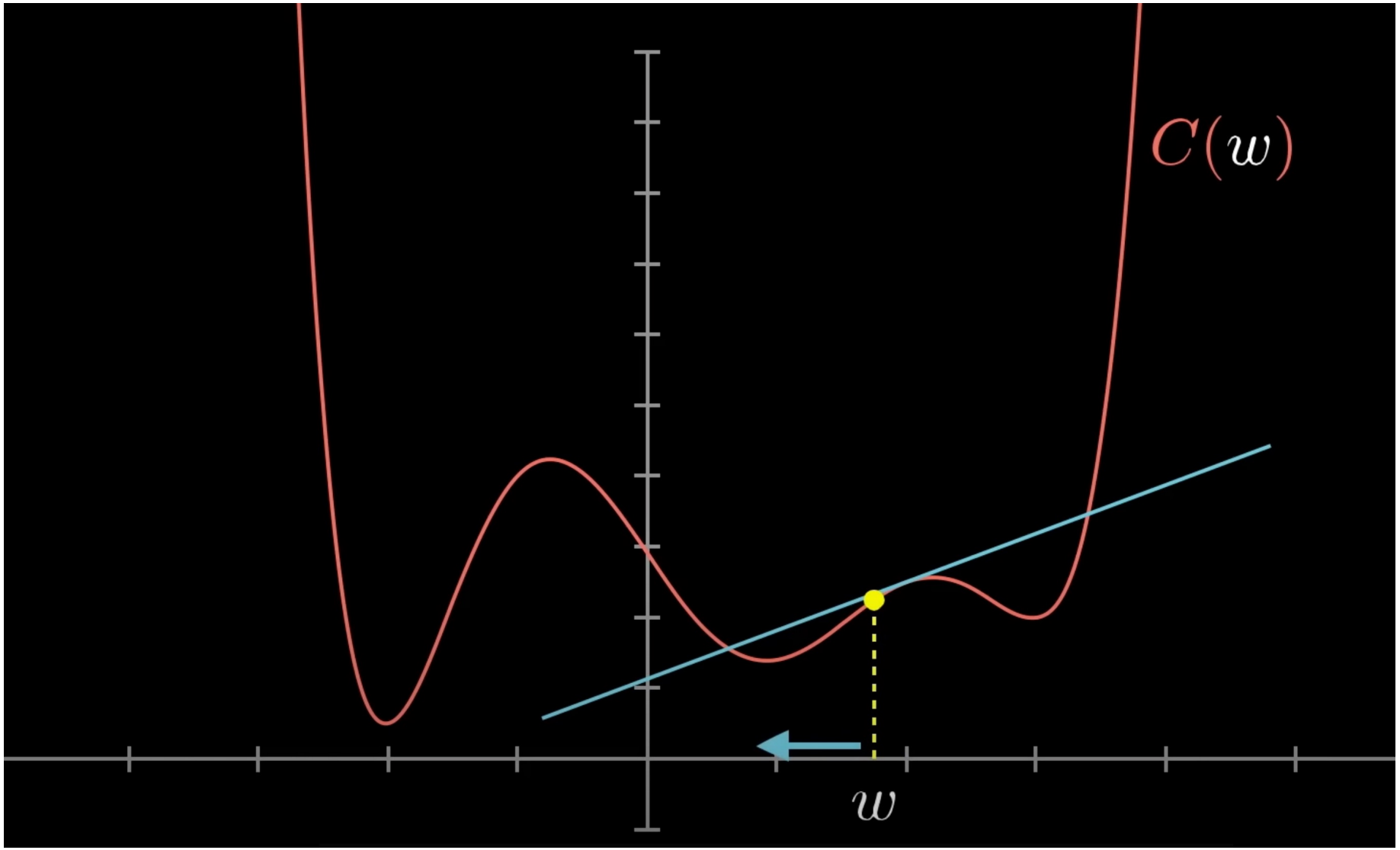
- Facilitates updating of weights
  - Metric of error: “loss function”
- What “direction” to move within function space of weights to minimize loss
- Calculus governs this
  - Outside the scope of this project/presentation


Loss
5
0
−5
−10
3
2
1
0
−1
−2
−3
Weights
−3
−2
−1
0
1
2
3
Weights
GOAL


$C(w)$
w


$C(w)$
w


$C(w)$
?
?
w


$C(w)$
w

Review Goals

- Build neural network
  - Learn and use PyTorch
  - Accurately predict redshift
  - Match accuracy of Dr. Singal's methods
- Optimize hyper-parameters
  - Learn and use Optuna
  - Find network structure that improves accuracy

Review Goals

- Build neural network
  - Learn and use PyTorch
  - Accurately predict redshift
  - Match accuracy of Dr. Singal's methods
- Optimize hyper-parameters
  - Learn and use Optuna
  - Find network structure that improves accuracy

WORK IN PROGRESS...

Data Dictionary

HSC

~ 290k rows

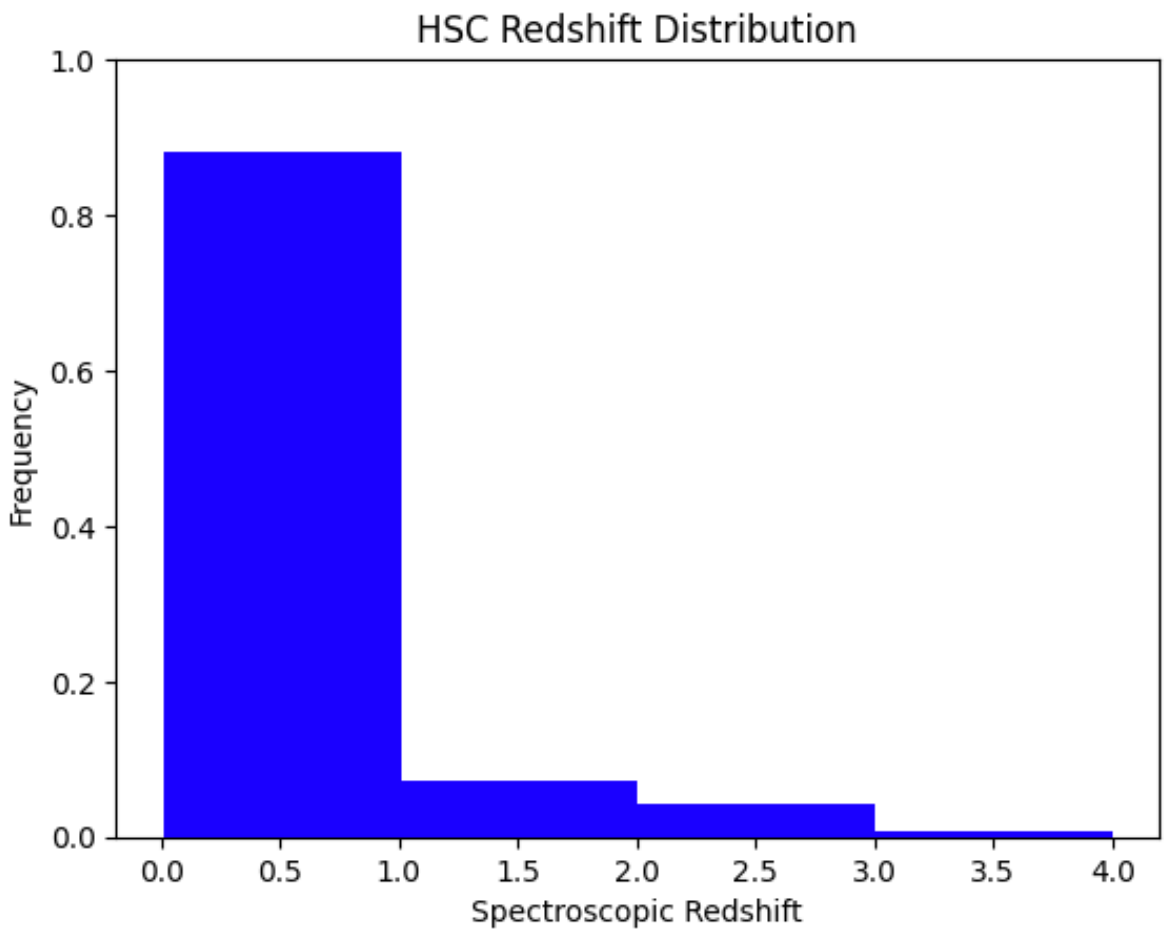
	A	B	C	D	E	F
1	u_mag	g_mag	r_mag	i_mag	zed_mag	z
2	25.426	25.567	25.2166	24.5832	24.0859	0.9966
3	26.0628	26.0293	25.419	25.1661	24.6804	1.8231
4	26.03	25.9201	24.8829	24.4989	24.438	0.5484
5	26.4852	26.0375	25.416	24.8233	24.2776	1.5998

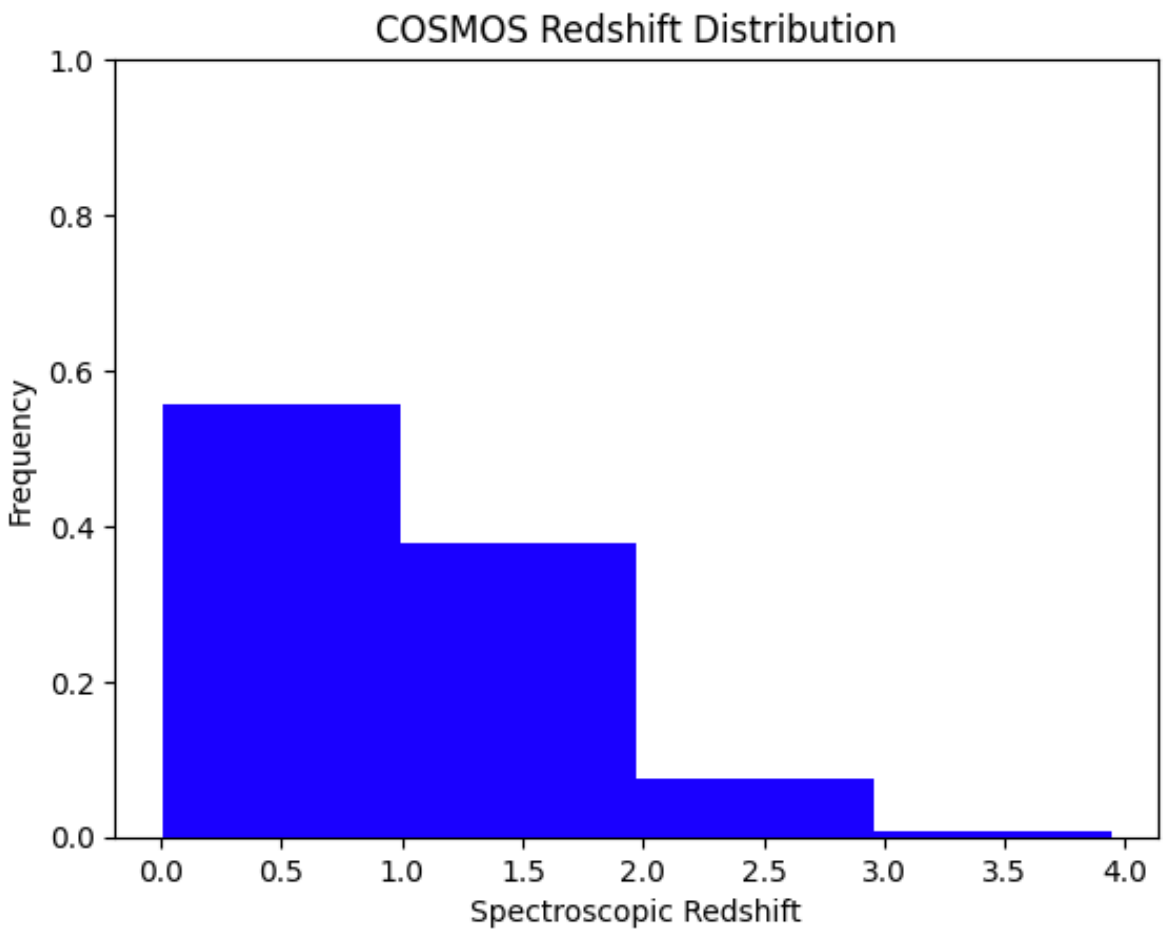
COSMOS2015

~ 60k rows

	A	B	C	D	E	F
1	g_cmodel_mag	r_cmodel_mag	i_cmodel_mag	z_cmodel_mag	y_cmodel_mag	specz_redshift
2	20.9795723	19.9471302	19.0677147	18.58988	18.398716	0.548910022
3	21.9359283	20.2798767	19.2970181	18.8702316	18.6586075	0.548210025
4	18.2886353	17.6349106	17.292387	17.095787	16.9314461	0.069250003
5	22.0716896	20.4420376	19.3423538	18.947113	18.7577057	0.565800011

Data Distribution





Density plots for scale

CODE

Import and Load Data

```python
import numpy as np
import pandas as pd
import torch
import torch.nn as nn
import torch.optim as optim
import matplotlib as plt
from torch.utils.data.dataset import random_split
from torch.utils.data import TensorDataset, DataLoader
from pathlib import Path
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error


# Load data
path = Path('/Users/katesautel/Documents/Seminar_2024/hsc_v6.csv')
#path = Path('/Users/katesautel/Documents/Seminar_2024/rel_z.csv')
df = pd.read_csv(path)

```

Prepare Data

```python
# Split data into features (F) and target (T)
F = df.iloc[:-1, 0:5]
T = df.iloc[:-1, 5]
```

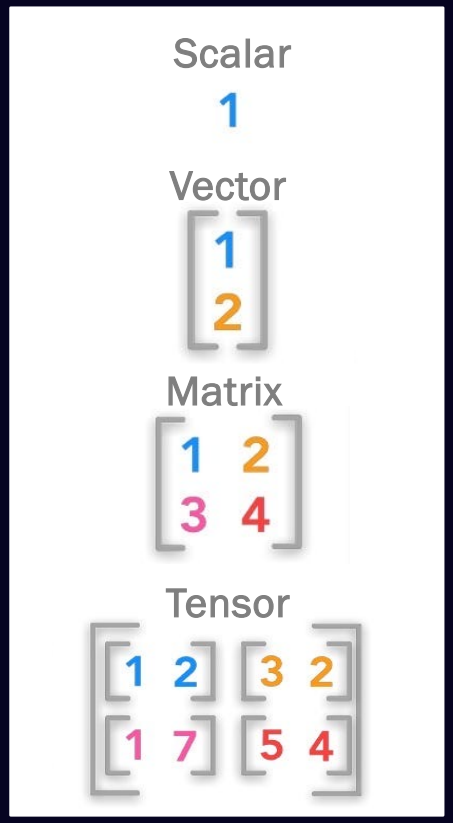
Select data

```python
# Scale features to be between 0 and 1
scaler = MinMaxScaler()
F_scaled = scaler.fit_transform(F)
```

Scale data

```python
# Convert data from arrays into tensors
F_tensor = torch.FloatTensor(F_scaled)
T_tensor = torch.FloatTensor(T.values)
```

Vector → tensor




```python
# Determine training vs. test set split (50%/50%)
train_size = len(data) // 2
test_size = len(data) - train_size

# Randomly split data into train and test
train_data, test_data = random_split(dataset = data, lengths = [train_size, test_size])
```

Train/test

Hyperparameters, Batch and Shuffle Data

```python

# Initialize hyperparameters
num_input_features = 5
num_hidden_neurons = 128
num_hidden_layers = 5
num_epochs = 300
learning_rate = 0.001
size_batch = 32
momentum = 0.9


# Create DataLoader objects (batch and shuffle data)
train_dl = DataLoader(dataset = train_data,
                      batch_size = size_batch,
                      shuffle = True)
test_dl = DataLoader(dataset = test_data,
                     batch_size = size_batch,
                     shuffle = False)

```

Shuffle data between iterations; parallel computation of batches

Create Neural Network

```python

# Define neural network
class NeuralNetwork(nn.Module):
    def __init__(self, num_input_features,
                 num_hidden_neurons,
                 num_hidden_layers):
        super(NeuralNetwork, self).__init__()
        self.input_layer = nn.Linear(num_input_features,
                                     num_hidden_neurons)
        self.hidden_layers = nn.ModuleList(
            [nn.Linear(num_hidden_neurons,
                       num_hidden_neurons) for n in range(num_hidden_layers)])
        self.output_layer = nn.Linear(num_hidden_neurons, 1)

    def forward(self, x):
        x = torch.relu(self.input_layer(x))
        for layer in self.hidden_layers:
            x = torch.relu(layer(x))
        x = self.output_layer(x)
        return x


# Initialize model
model = NeuralNetwork(num_input_features, num_hidden_neurons, num_hidden_layers)

```

Class definition jargon

Bulk of NN

Train the Model

```python

# Define loss function and optimizer
criterion = nn.MSELoss()
optimizer = optim.SGD(model.parameters(), lr=learning_rate, momentum=momentum)

min_loss = 1
best_model = None
loss_data = []


# Train the model
for epoch in range(num_epochs):
    for inputs, targets in train_dl:
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, targets.unsqueeze(1))
        loss.backward()
        optimizer.step()

    if loss.item() < min_loss:
        min_loss = loss.item()
        best_model = model.state_dict()

    if (epoch + 1) % 1 == 0:  # Print every epoch
        print(f'Epoch [{epoch + 1}/{num_epochs}], Loss: {loss.item():.4f}')
        loss_data.append(loss.item())
        print()

```

Print/save loss data each iteration

Save Best Model and Evaluate Performance

```python

# Save the best model
if best_model is not None:
    torch.save(best_model, '/Users/katesautel/Documents/Seminar_2024/best_model.pth')


# Load best model
best_model = NeuralNetwork(num_input_features, num_hidden_neurons, num_hidden_layers)
best_model.load_state_dict(torch.load('/Users/katesautel/Documents/Seminar_2024/best_model.pth'))


# Evaluate model performance
best_model.eval()
with torch.no_grad():
    y_true = []
    y_pred = []

    for inputs, targets in test_dl:
        outputs = best_model(inputs)

        mse = mean_squared_error(targets.numpy(), outputs.numpy().squeeze())

        y_true.extend(targets.numpy())
        y_pred.extend(outputs.numpy().squeeze())

```

Find/save/load model with best loss

★ LOSS FUNCTION

Categorize Outliers and Save Data

```python

# Save the results
y_true = np.array(y_true)
y_pred = np.array(y_pred)
results = pd.DataFrame({'Spectroscopic': y_true, 'Photometric': y_pred})


# Categorize outliers
results['Outlier'] = np.select([abs(results['Photometric'] -
                                    results['Spectroscopic']) > 1,
                                abs(results['Photometric'] -
                                    results['Spectroscopic']) > 0.15 * (1 + results['Spectroscopic'])],
                               ['CO', 'O'], 'NO')


# Export data
results.to_csv('/Users/katesautel/Documents/Seminar_2024/results.csv', index=False)

loss_data = pd.DataFrame({'Loss': loss_data})
loss_data.to_csv('/Users/katesautel/Documents/Seminar_2024/loss.csv', index=False)

print("MSE:", mse)
```

Uses equations from beginning to categorize

Save as CSV

Prints MSE for entire run

Create Loss Plot

```python
# Import libraries
import pandas as pd
import numpy as np
import matplotlib as plt
import matplotlib.pyplot as plt


# Load data
data = pd.read_csv('/Users/katesautel/Documents/Seminar_2024/results.csv')
dfData = pd.DataFrame(data)

loss = pd.read_csv('/Users/katesautel/Documents/Seminar_2024/loss.csv')
dfLoss = pd.DataFrame(loss)


# LOSS PLOT
dfLoss['MA'] = dfLoss['Loss'].rolling(10).mean()

fig, ax = plt.subplots()

ax.plot(dfLoss['Loss'],
        c = 'blue',
        ls = 'solid',
        lw = 1)
ax.plot(dfLoss['MA'],
        c = 'red',
        ls = 'solid',
        lw = 2)

ax.set_title('Loss vs. Epoch')
ax.set_xlabel('Epoch Number')
ax.set_ylabel('Loss')
ax.set_ylim(0, 1)
ax.legend(['Loss', '10-Epoch MA'])

fig.show()
```

Create Photo vs. Spectro Scatter Plot

```python
# REGRESSION PLOT
fig, ax = plt.subplots()

ax.scatter(dfData['Spectroscopic'], dfData['Photometric'],
           c = 'blue',
           s = 1,
           alpha = 0.5)

# Outlier guide lines
x1_values = np.linspace(0, 5, 100)
y1_values = 1.15 * x1_values + 0.15
ax.plot(x1_values, y1_values,
        c = 'black',
        ls = 'dashed',
        lw = 1)

x2_values = np.linspace(0, 5, 100)
y2_values = 0.85 * x2_values - 0.15
ax.plot(x2_values, y2_values,
        c = 'black',
        ls = 'dashed',
        lw = 1)

# CO guide lines
x3_values = np.linspace(0, 5, 100)
y3_values = 1 + x3_values
ax.plot(x3_values, y3_values,
        c = 'black',
        ls = 'solid',
        lw = 1)

x4_values = np.linspace(0, 5, 100)
y4_values = x4_values - 1
ax.plot(x4_values, y4_values,
        c = 'black',
        ls = 'solid',
        lw = 1)

ax.set_title('Photometric vs. Spectroscopic Redshift')
ax.set_xlabel('Spectroscopic (Actual)')
ax.set_ylabel('Photometric (Estimated)')
ax.set_xlim(0, 4)
ax.set_ylim(0, 4)
#ax.legend(['Trial', 'O', 'O', 'CO', 'CO'])
ax.text(2, 3.5, 'CO', fontsize = 15)
ax.text(3.5, 2, 'CO', fontsize = 15)
ax.text(2.6, 3.35, 'O', fontsize = 15)
ax.text(3.4, 2.55, 'O', fontsize = 15)
ax.text(3, 3, 'NO', fontsize = 15)

fig.show()
```

$$O: \frac{\left|z_{\text{phot}} - z_{\text{spec}}\right|}{1 + z_{\text{spec}}} > 0.15$$

$$O_c: \left|z_{\text{phot}} - z_{\text{spec}}\right| > 1.0$$

Create Stats Chart

```
# Print results stats
stats = pd.DataFrame()
stats['Outlier Counts'] = data['Outlier'].value_counts()
stats['Proportion'] = stats['Outlier Counts'] / stats['Outlier Counts'].sum()
stats
```

✓ 0.0s

RESULTS

Raw Results

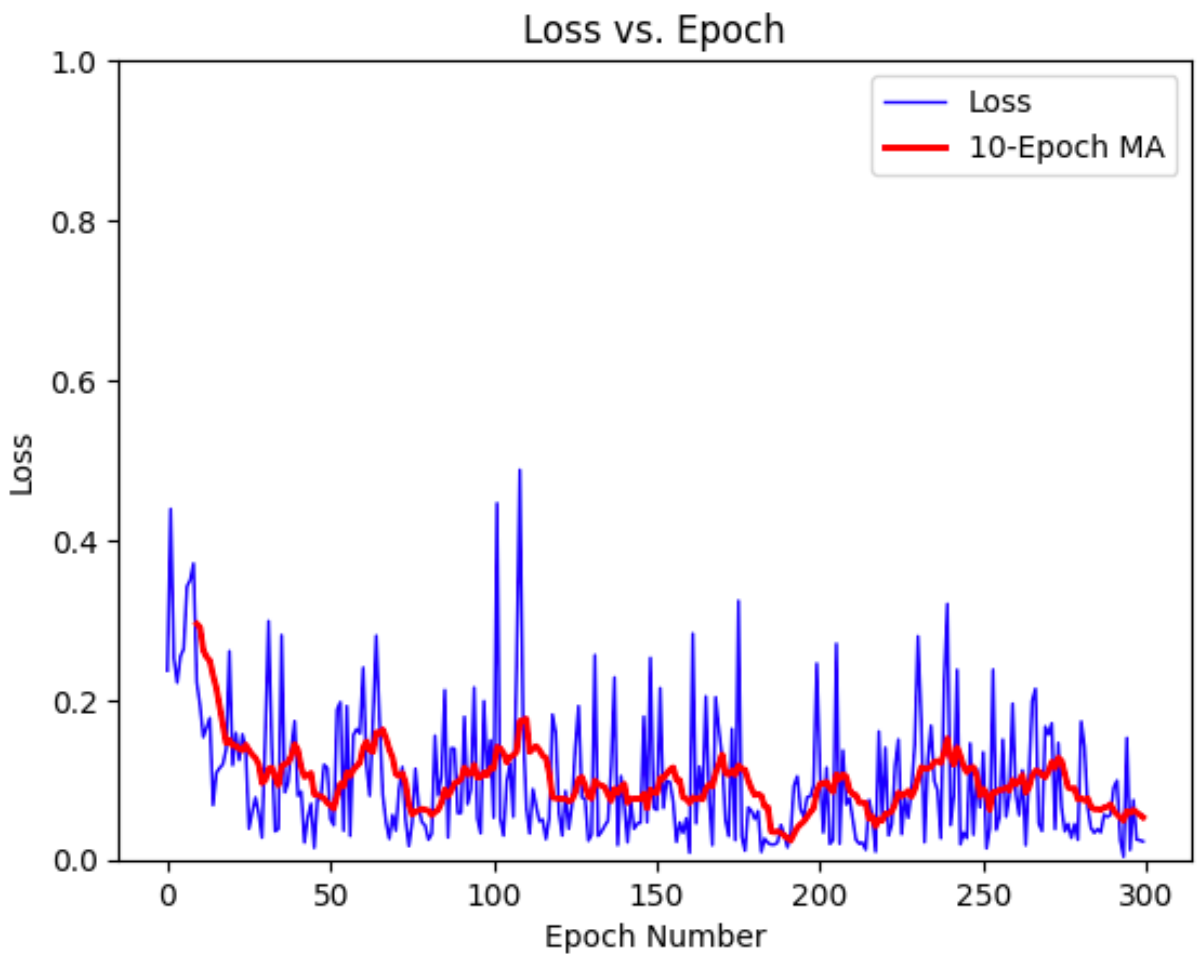
	A	B	C
1	Spectroscopic	Photometric	Outlier
2	0.54582	0.5754829	NO
3	0.57347	0.6121583	NO
4	0.78753	0.67352504	NO
5	0.32376	0.2998501	NO
6	2.56503	2.4574053	NO
7	0.22696	0.23847589	NO
8	2.0192	1.6988337	NO
9	0.13027	0.08969337	NO
10	0.10591	0.11179638	NO

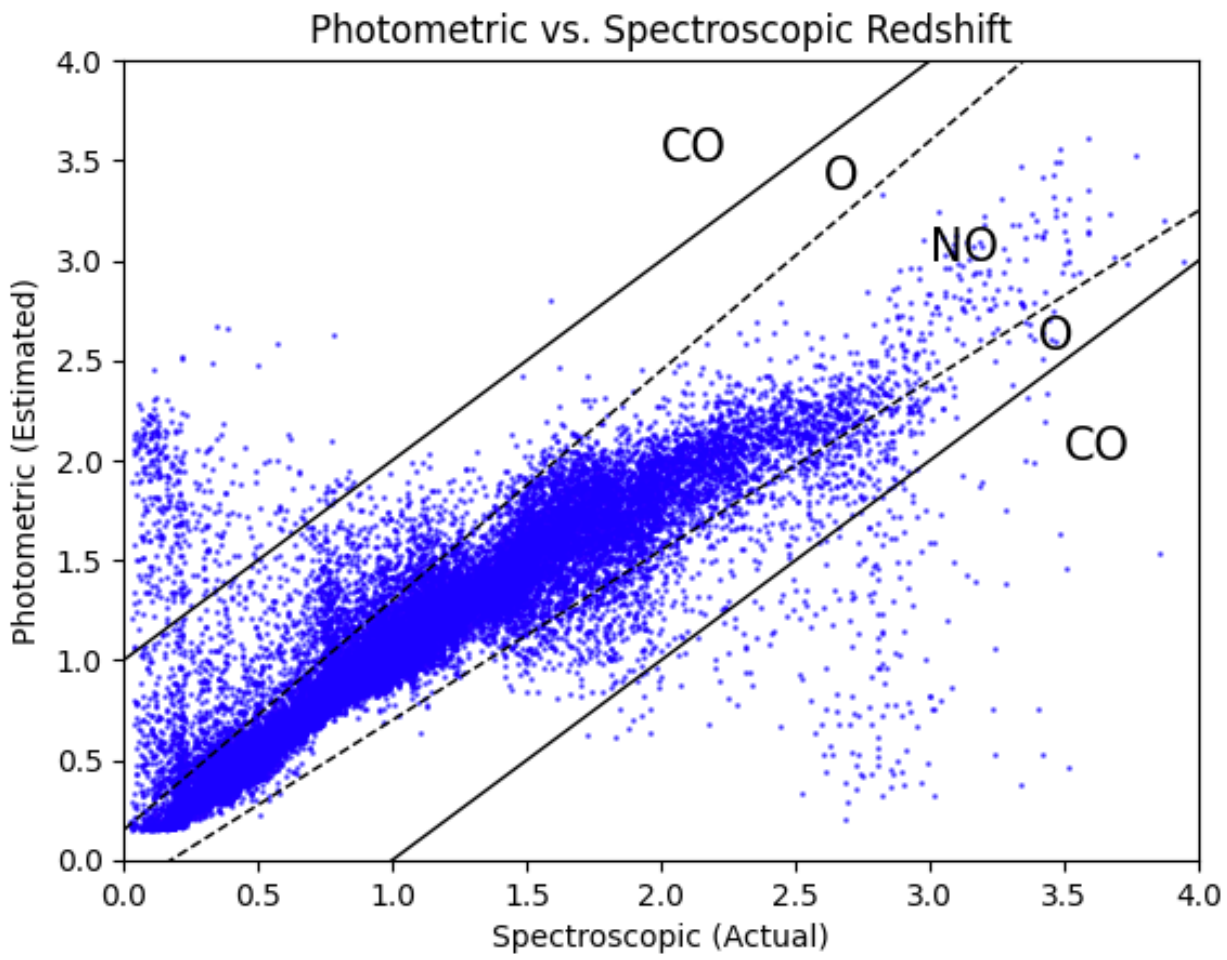
	A
1	Loss
2	0.16382750868797302
3	0.024627924
4	0.060656987
5	0.12787343561649323
6	0.096915103
7	0.1874314546585083
8	0.037416216
9	0.057485376
10	0.13563856482505798

HSC Dataset

~ 290k, 30 min processing time

Outlier	Outlier Counts	Proportion
NO	25976	0.886281
O	2727	0.093043
CO	606	0.020676

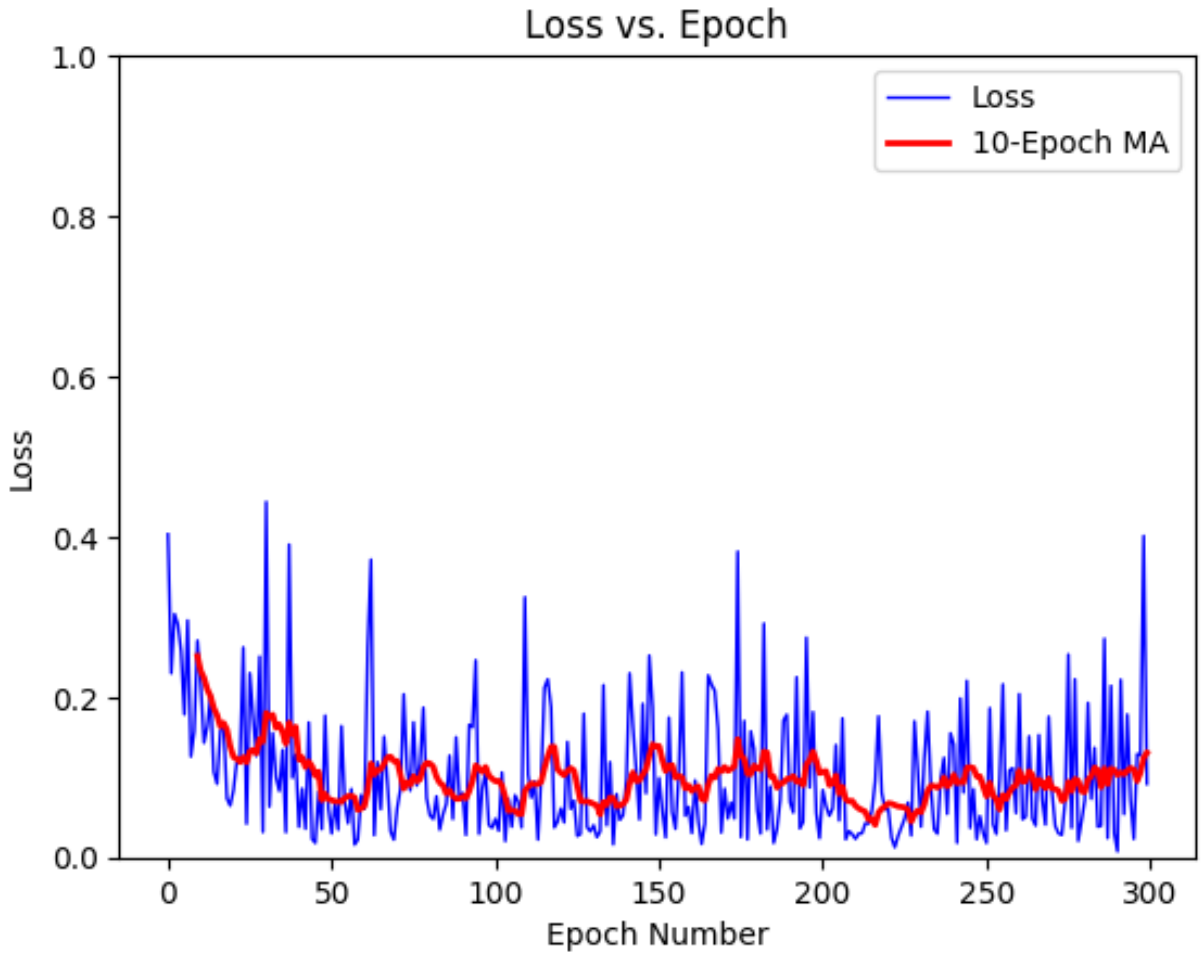


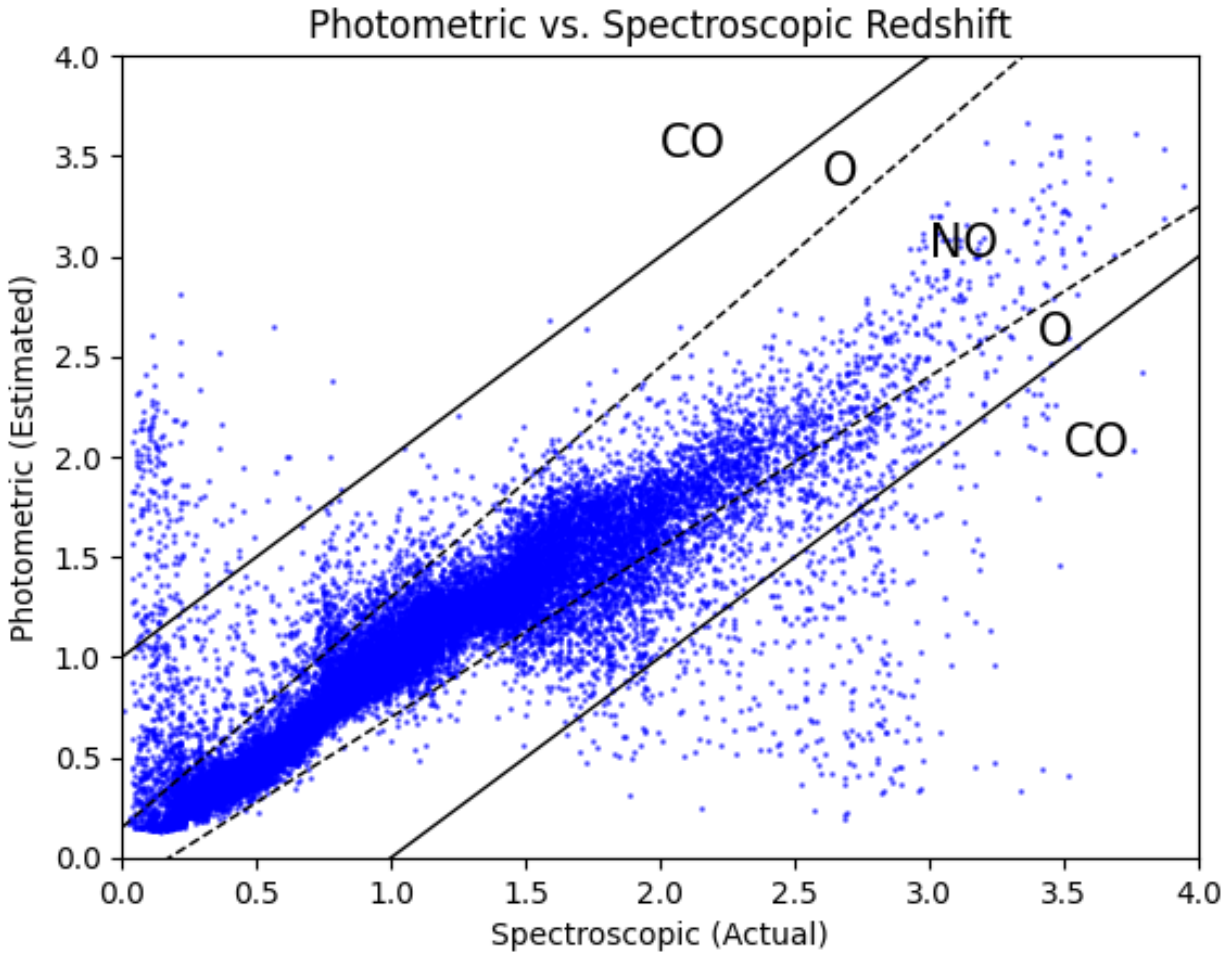


COSMOS Dataset

~ 60k, 7 min processing time

Outlier	Outlier Counts	Proportion
NO	26252	0.895698
O	2473	0.084377
CO	584	0.019926





Optuna

- Hyperparameter optimization framework
- Curious about best network structure
  - Hidden layers, hidden neurons

1. Wrap around existing model
2. Quickly and efficiently tests values of hyperparameters
3. Finds best combination of hyperparameters

Hyperparameters: properties of NN

- Hidden layers
- Neurons per layer
- Loss metric
- Momentum
- Learning rate
- Epochs
- Batch size

```python
# Create objective function for Optuna
def objective(trial):
    # Define hyperparameters to optimize
    num_hidden_neurons = trial.suggest_int('num_hidden_neurons', 16, 256)
    num_hidden_layers = trial.suggest_int('num_hidden_layers', 1, 5)
    #learning_rate = trial.suggest_float('learning_rate', 0.00001, 0.1, log = True)
    #num_epochs = trial.suggest_int('num_epochs', 0, 500)
    #momentum = trial.suggest_float('momentum', 0, 1)

    # Initialize model
    model = NeuralNetwork(num_input_features = 5,
                          num_hidden_neurons = num_hidden_neurons,
                          num_hidden_layers = num_hidden_layers)

    # Define loss function and optimizer
    criterion = nn.MSELoss()
    optimizer = optim.SGD(model.parameters(), lr = learning_rate)

    # Train the model
    for epoch in range(num_epochs):
        for inputs, targets in train_dl:
            optimizer.zero_grad()
            outputs = model(inputs)
```

```python
# Optuna study
study = optuna.create_study(direction = 'minimize')
study.optimize(objective, n_trials = 100)


# Get best hyperparameters
best_params = study.best_params
num_hidden_neurons = best_params['num_hidden_neurons']
num_hidden_layers = best_params['num_hidden_layers']


# Train model with best hyperparameters
model = NeuralNetwork(num_input_features = 5,
                      num_hidden_neurons = num_hidden_neurons,
                      num_hidden_layers = num_hidden_layers)
criterion = nn.MSELoss()
optimizer = optim.SGD(model.parameters(), lr = learning_rate)

for epoch in range(num_epochs):
```

Optuna Status

(As of 2/27)

- Moving to Google Colab
  - Too many issues downloading Optuna on my computer
  - GPU > CPU
    - Parallel computing capabilities

```
A new study created in memory with name: no-name-5bb9c40c-ed7a-41bb-b24e-70bd869aaff8
Trial 0 finished with value: 0.12018265575170517 and parameters: {'num_hidden_neurons': 121, 'num_hidden_layers': 4}. Best is trial 0 with value: 0.12018265575170517.
Trial 1 finished with value: 0.12069696187973022 and parameters: {'num_hidden_neurons': 241, 'num_hidden_layers': 4}. Best is trial 0 with value: 0.12018265575170517.
Trial 2 finished with value: 0.1291436403989792 and parameters: {'num_hidden_neurons': 117, 'num_hidden_layers': 2}. Best is trial 0 with value: 0.12018265575170517.
Trial 3 finished with value: 0.12597191333770752 and parameters: {'num_hidden_neurons': 63, 'num_hidden_layers': 2}. Best is trial 0 with value: 0.12018265575170517.
Trial 4 finished with value: 0.12624286115169525 and parameters: {'num_hidden_neurons': 163, 'num_hidden_layers': 2}. Best is trial 0 with value: 0.12018265575170517.
Trial 5 finished with value: 0.12193232774734497 and parameters: {'num_hidden_neurons': 55, 'num_hidden_layers': 4}. Best is trial 0 with value: 0.12018265575170517.
Trial 6 finished with value: 0.12263752520084381 and parameters: {'num_hidden_neurons': 80, 'num_hidden_layers': 2}. Best is trial 0 with value: 0.12018265575170517.
Trial 7 finished with value: 0.1315770298242569 and parameters: {'num_hidden_neurons': 66, 'num_hidden_layers': 4}. Best is trial 0 with value: 0.12018265575170517.
Trial 8 finished with value: 0.12338316440582275 and parameters: {'num_hidden_neurons': 189, 'num_hidden_layers': 2}. Best is trial 0 with value: 0.12018265575170517.
```

Next Steps

- Metric of success
  - Minimize COs?
  - Minimize loss (MSE/RMS)
- Results will be in paper but not complete at this point

Sources

- https://en.wikipedia.org/wiki/Photometry_(astronomy)
- https://en.wikipedia.org/wiki/Photometric_redshift
- https://www.superannotate.com/blog/guide-to-gradient-descent-algorithms
- https://www.youtube.com/watch?v=IHZwWFHWa-w&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi&index=2
- https://optuna.org
- https://www.coursera.org/articles/ai-vs-deep-learning-vs-machine-learning-beginners-guide
- https://www.youtube.com/watch?v=aircAruvnKk

Thank You, Dr. Singal!

Questions?