

HW5 Yaroslav Popryho
ypopry2@uic.edu

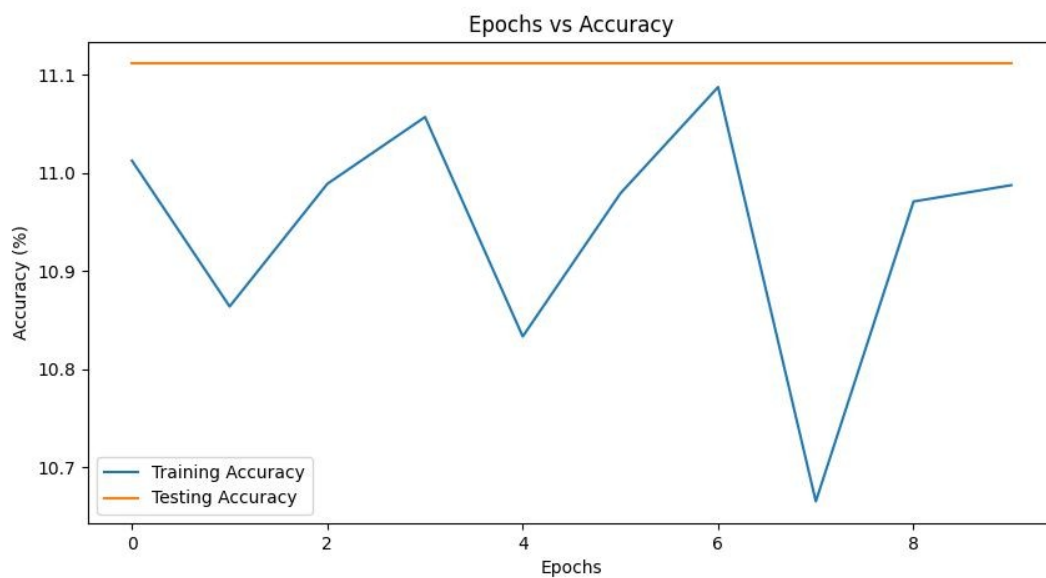
- The details of your neural network design process, what did you try, what worked, what did not... The optimizer, the loss function, and other hyperparameters that you utilized.

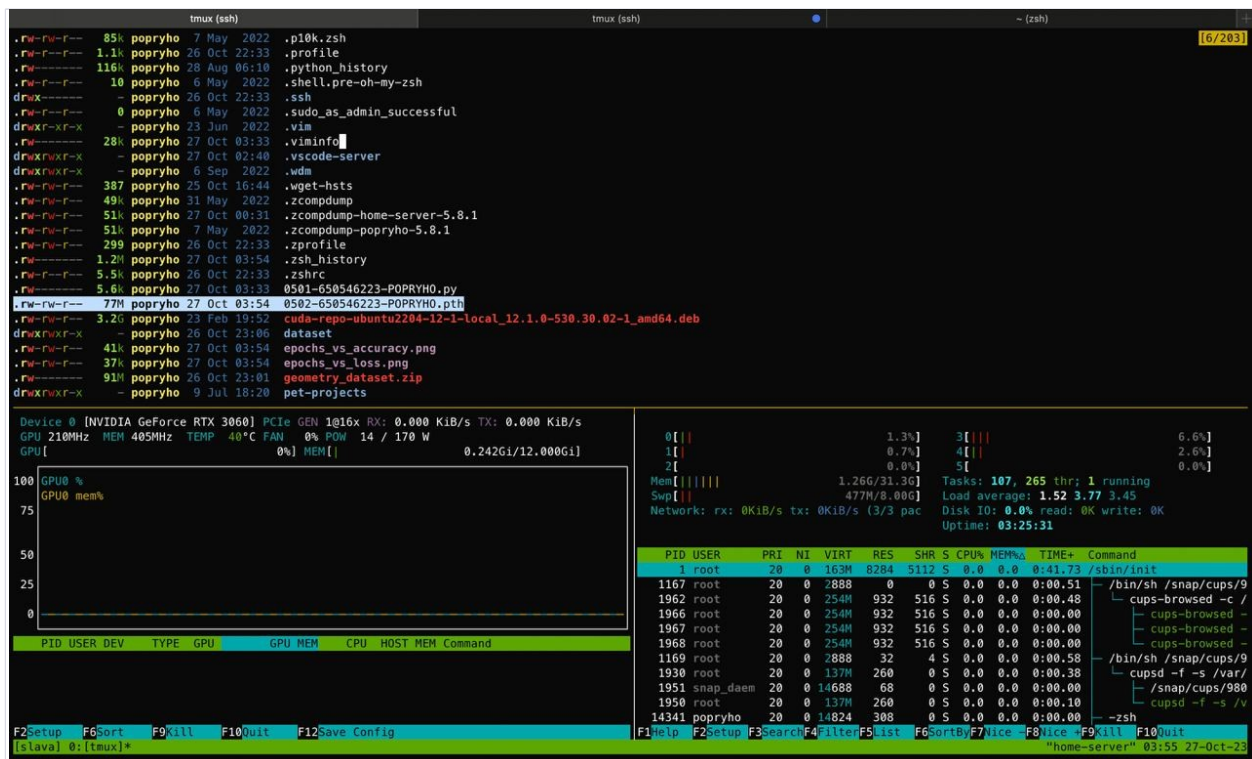
In the first experiment, I utilized the following architecture:

- 4 convolutional layers with channels increasing from 32 to 256.
- 3 max-pooling layers.
- 2 fully connected layers, with a dropout of 50% before the final layer.

Hyper parameters:

- Learning rate: 0.001.
- Batch size: 132.
- Optimizer: Adam
- Loss function: Cross Entropy Loss



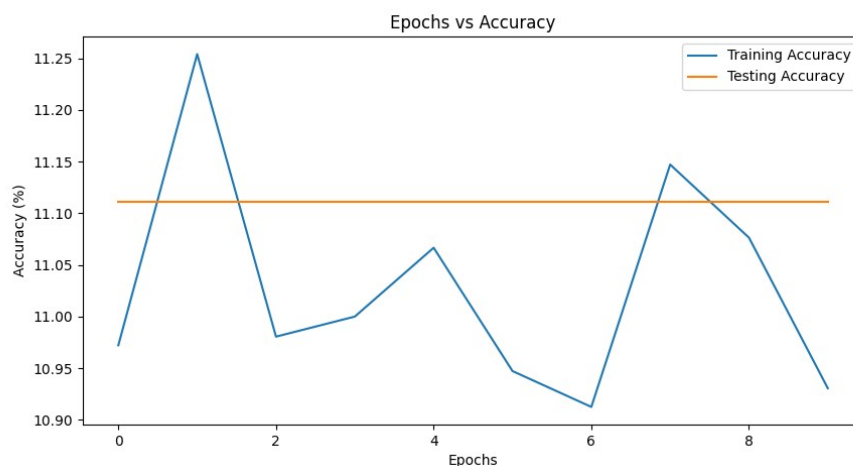


Result: The loss on the test set was higher than on the training set, indicating overfitting. Surprisingly, the training accuracy was also lower than expected. Additionally, the model size (77 MB) exceeded the acceptable limit.

For the 2nd experiment, I was using the same model architecture from experiment 1, but just modified the training parameters:

- Set learning rate to 0.01.

Result: The testing accuracy remained constant across epochs, suggesting that the model wasn't learning effectively or had reached a local minimum.



I then opted for a simpler model with following changes:

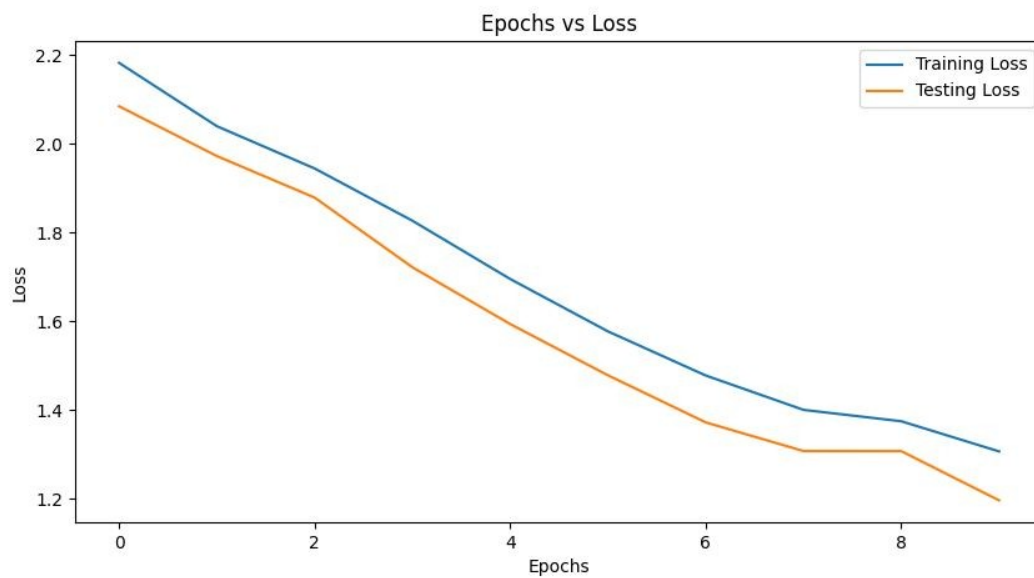
- Reduced the number of channels in convolutional layers.
- Removed one convolutional layer.
- Adjusted the size of the fully connected layers.

Hyper parameters:

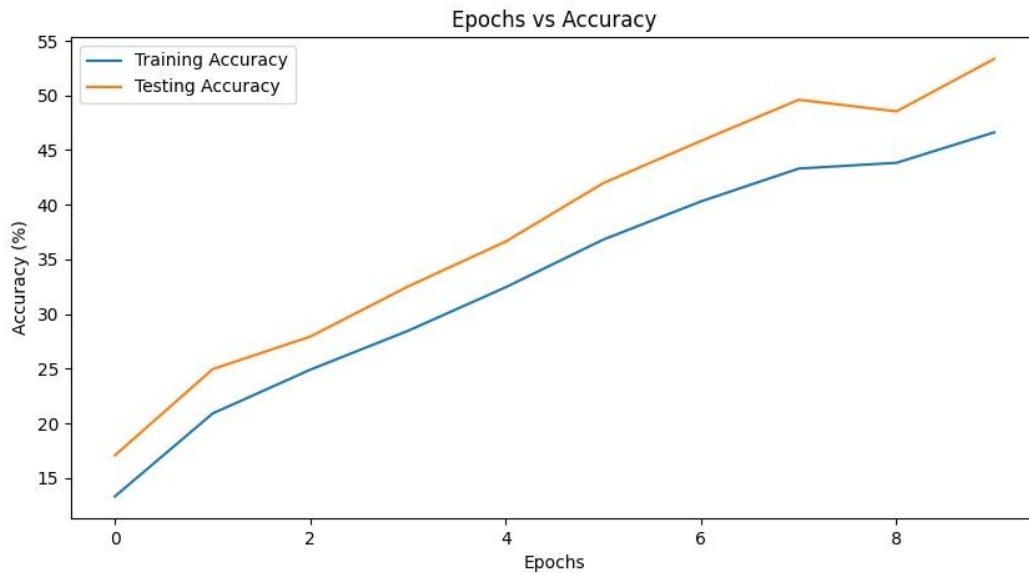
- Batch size: 32.
- Learning rate: 0.001.

Result: This model showed the expected behavior with increasing accuracy and decreasing loss across epochs for both training and testing sets, signifying improved generalization and performance.

- A graph that shows epochs vs loss on training set, and on the same graph epoch vs loss on the set set.



- A graph that shows epochs vs accuracy on training set, and on the same graph epoch vs accuracy on the set set. - All your codes



- All your codes.

0501-650546223-POPRYHO.py:

```

HW5 > 0501-650546223-POPRYHO.py > GeometryCNN > forward
1 import os
2 import torch
3 import matplotlib.pyplot as plt
4 from PIL import Image
5 from torchvision import transforms
6 from torch.utils.data import DataLoader, Dataset, TensorDataset
7 import torch.nn as nn
8 import torch.nn.functional as F
9 import torch.optim as optim
10 from tqdm import tqdm
11
12 device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
13
14 class GeometryCNN(nn.Module):
15     def __init__(self, num_classes=9):
16         super(GeometryCNN, self).__init__()
17
18         self.conv1 = nn.Conv2d(in_channels=3, out_channels=16, kernel_size=3, stride=2, padding=1) # Output: 100x100
19         self.conv2 = nn.Conv2d(16, 32, kernel_size=3, padding=1) # Output: 50x50 (due to pooling)
20         self.conv3 = nn.Conv2d(32, 64, kernel_size=3, padding=1) # Output: 25x25 (due to pooling)
21
22         self.pool = nn.MaxPool2d(kernel_size=2, stride=2, padding=0)
23
24         self.fc1 = nn.Linear(64 * 25 * 25, 128)
25         self.fc2 = nn.Linear(128, num_classes)
26
27         self.dropout = nn.Dropout(0.5)
28
29     def forward(self, x):
30         x = F.relu(self.conv1(x))
31
32         x = self.pool(F.relu(self.conv2(x)))
33         x = self.pool(F.relu(self.conv3(x)))
34         x = x.view(-1, 64 * 25 * 25)
35         x = F.relu(self.fc1(x))
36         x = self.dropout(x)
37         x = self.fc2(x)
38
39         return x
40
41 class GeometryDataset(Dataset):
42     def __init__(self, image_files, labels, transform=None):
43         self.image_files = image_files
44         self.labels = labels
45         self.transform = transform
46         self.classes = list(set(labels))

```

> HW1

> HW2

> HW3

> HW4

> venv

04-650546223-P...

Figure_1.png

Figure_2.png

Figure_3.png

HW4 Yaroslav Popr...

HW4 Yaroslav Popr...

> HW5

> output

> venv

.-lock.HW5 Yarosl...

0501-650546... 1

0502-650546223-...

0503-650546... 1

geometry_dataset....

HW5 Yaroslav Popr...

hw5.pdf

test.py 1

> TIMELINE

```
40
41
42 class GeometryDataset(Dataset):
43     def __init__(self, image_files, labels, transform=None):
44         self.image_files = image_files
45         self.labels = labels
46         self.transform = transform
47         self.classes = list(set(labels))
48
49     def __len__(self):
50         return len(self.image_files)
51
52     def __getitem__(self, idx):
53         img_name = os.path.join(DATA_DIR, self.image_files[idx])
54         image = Image.open(img_name)
55         label = self.classes.index(self.labels[idx])
56
57         if self.transform:
58             image = self.transform(image)
59
60         return image, label
61
62 DATA_DIR = '/Users/yaroslavpopryho/Study/UIC/Classes/Neural Networks/HW5/output'
63
64 transform = transforms.Compose([transforms.ToTensor()])
65
66 all_files = os.listdir(DATA_DIR)
67 all_images = [f for f in all_files if f.endswith('.png')]
68 labels = [f.split('_')[0] for f in all_images]
69
70 train_images, test_images = [], []
71 train_labels, test_labels = [], []
72
73 classes = list(set(labels))
74 for cls in classes:
75     class_images = [img for img, label in zip(all_images, labels) if label == cls]
76     class_images.sort()
77     train_images.extend(class_images[:8000])
78     test_images.extend(class_images[8000:])
79     train_labels.extend([cls] * 8000)
80     test_labels.extend([cls] * 2000)
81
82 train_dataset = GeometryDataset(train_images, train_labels, transform=transform)
83 test_dataset = GeometryDataset(test_images, test_labels, transform=transform)
84
85 # Hyperparameters
86 batch_size = 32
```

NEURAL NETWORKS

> HW1

> HW2

> HW3

> HW4

> venv

04-650546223-P...

Figure_1.png

Figure_2.png

Figure_3.png

HW4 Yaroslav Popr...

HW4 Yaroslav Popr...

HW5

> output

> venv

lock.HW5 Yarosl...

0501-650546... 1

0502-650546223-...

0503-650546... 1

geometry_dataset....

HW5 Yaroslav Popr...

hw5.pdf

test.py 1

TIMELINE

HW5 > 0501-650546223-POPRYHO.py > GeometryCNN > forward

86 learning_rate = 0.01

87 epochs = 10

88

89

90 train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)

91 test_loader = DataLoader(test_dataset, batch_size=batch_size, shuffle=False)

92

93 model = GeometryCNN().to(device)

94 criterion = nn.CrossEntropyLoss()

95 optimizer = optim.Adam(model.parameters(), lr=learning_rate)

96

97 train_losses, test_losses = [], []

98 train_accuracies, test_accuracies = [], []

99

100 for epoch in tqdm(range(epochs), desc="Epochs"):

101 # Training

102 model.train()

103 total_train_loss = 0

104 total_train_correct = 0

105

106 with tqdm(train_loader, desc="Train Batch", leave=False) as pbar:

107 for batch_idx, (inputs, labels) in enumerate(pbar):

108 inputs, labels = inputs.to(device), labels.to(device)

109 optimizer.zero_grad()

110 outputs = model(inputs)

111 loss = criterion(outputs, labels)

112 loss.backward()

113 optimizer.step()

114

115 total_train_loss += loss.item()

116 pbar.set_description(f"Epoch {epoch+1} Train Loss: {total_train_loss/(batch_idx+1):.4f}")

117

118 _, predicted = torch.max(outputs.data, 1)

119 total_train_correct += (predicted == labels).sum().item()

120

121 train_losses.append(total_train_loss / len(train_loader))

122 train_accuracies.append(100 * total_train_correct / len(train_dataset))

123

124 # Evaluation

125 model.eval()

126 total_test_loss = 0

127 total_test_correct = 0

128

129 with tqdm(test_loader, desc="Test Batch", leave=False) as pbar:

130 for batch_idx, (inputs, labels) in enumerate(pbar):

131

> HW1

> HW2

> HW3

> HW4

> venv

04-650546223-P...

Figure_1.png

Figure_2.png

Figure_3.png

HW4 Yaroslav Popr...

HW4 Yaroslav Popr...

HW5

> output

> venv

~lock.HW5 Yarosl...

0501-650546... 1

0502-650546223-...

0503-650546... 1

geometry_dataset....

HW5 Yaroslav Popr...

hw5.pdf

test.py 1

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

```
train_accuracies.append(100 * total_train_correct / len(train_dataset))

# Evaluation
model.eval()
total_test_loss = 0
total_test_correct = 0

with tqdm(test_loader, desc="Test Batch", leave=False) as pbar:
    for batch_idx, (inputs, labels) in enumerate(pbar):

        inputs, labels = inputs.to(device), labels.to(device)
        outputs = model(inputs)
        loss = criterion(outputs, labels)

        total_test_loss += loss.item()
        pbar.set_description(f"Epoch {epoch+1} Test Loss: {total_test_loss/(batch_idx+1):.4f}")

        _, predicted = torch.max(outputs.data, 1)
        total_test_correct += (predicted == labels).sum().item()

test_losses.append(total_test_loss / len(test_loader))
test_accuracies.append(100 * total_test_correct / len(test_dataset))

MODEL_PATH = "0502-650546223-POPRYHO.pth"
torch.save(model.state_dict(), MODEL_PATH)

# Epochs vs Loss
plt.figure(figsize=(10, 5))
plt.plot(train_losses, label='Training Loss')
plt.plot(test_losses, label='Testing Loss')
plt.title('Epochs vs Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.savefig('epochs_vs_loss.png')

# Epochs vs Accuracy
plt.figure(figsize=(10, 5))
plt.plot(train_accuracies, label='Training Accuracy')
plt.plot(test_accuracies, label='Testing Accuracy')
plt.title('Epochs vs Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy (%)')
plt.legend()
plt.savefig('epochs_vs_accuracy.png')
```

TIMELINE

0503-650546223-POPRYHO.py

HW5 > 0503-650546223-POPRYHO.py > GeometryCNN > forward

```
1 import os
2 from PIL import Image
3 import torch
4 from torchvision import transforms
5 import torch.nn as nn
6 from torch.utils.data import DataLoader, Dataset
7 import torch.nn.functional as F
8
9 class GeometryCNN(nn.Module):
10     def __init__(self, num_classes=9):
11         super(GeometryCNN, self).__init__()
12
13         self.conv1 = nn.Conv2d(in_channels=3, out_channels=16, kernel_size=3, stride=2, padding=1) # Output: 100x100
14         self.conv2 = nn.Conv2d(16, 32, kernel_size=3, padding=1) # Output: 50x50 (due to pooling)
15         self.conv3 = nn.Conv2d(32, 64, kernel_size=3, padding=1) # Output: 25x25 (due to pooling)
16
17         self.pool = nn.MaxPool2d(kernel_size=2, stride=2, padding=0)
18
19         self.fc1 = nn.Linear(64 * 25 * 25, 128)
20         self.fc2 = nn.Linear(128, num_classes)
21
22         self.dropout = nn.Dropout(0.5)
23
24     def forward(self, x):
25         x = F.relu(self.conv1(x))
26
27         x = self.pool(F.relu(self.conv2(x)))
28         x = self.pool(F.relu(self.conv3(x)))
29         x = x.view(-1, 64 * 25 * 25)
30         x = F.relu(self.fc1(x))
31         x = self.dropout(x)
32         x = self.fc2(x)
33
34         return x
35
36
37 class GeometryInferenceDataset(Dataset):
38     def __init__(self, root_dir, transform=None):
39         self.root_dir = root_dir
40         self.image_files = [f for f in os.listdir(root_dir) if f.endswith('.png')]
41         self.transform = transform
42
43     def __len__(self):
44         return len(self.image_files)
45
46     def __getitem__(self, idx):
```



```

self.root_dir = root_dir
self.image_files = [f for f in os.listdir(root_dir) if f.endswith('.png')]
self.transform = transform

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

def __getitem__(self, idx):
    img_name = os.path.join(self.root_dir, self.image_files[idx])
    image = Image.open(img_name)
    if self.transform:
        image = self.transform(image)
    return image, self.image_files[idx]

device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")

model = GeometryCNN().to(device)
model.load_state_dict(torch.load("0502-650546223-POPRYHO.pth", map_location=device))
model.eval()

transform = transforms.Compose([transforms.ToTensor()])

dataset = GeometryInferenceDataset(root_dir="output/", transform=transform)

dataloader = DataLoader(dataset, batch_size=32, shuffle=False)

classes = ["Circle", "Square", "Octagon", "Heptagon", "Nonagon", "Star", "Hexagon", "Pentagon", "Triangle"]

for images, filenames in dataloader:
    images = images.to(device)
    outputs = model(images)
    predictions = outputs.argmax(dim=1)
    for filename, prediction in zip(filenames, predictions):
        print(f"{filename}: {classes[prediction.item()]}" )

```

