1a) Andrew Onozuka A16760043

Problem 1

b)
$\alpha_0 = A$
$\alpha_1 = 1$
$\alpha_2 = 6$
$\alpha_3 = 7$
$\alpha_4 = 6$
$\alpha_5 = 0$
$\alpha_6 = 0$
$\alpha_7 = 4$
$\alpha_8 = 3$

c)
email sent

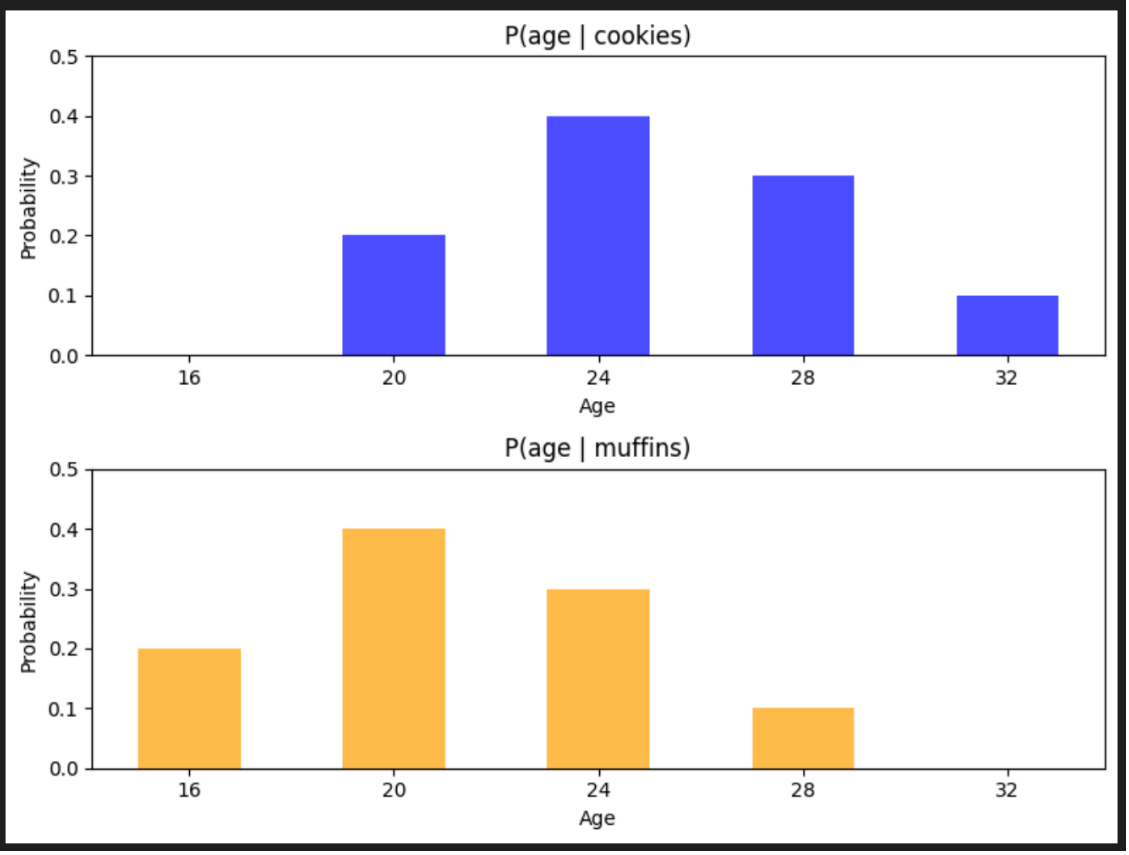
1a) Andrew Onozuka A16760043

Problem 2

a)

```python
import matplotlib.pyplot as plt

ages = [16, 20, 24, 28, 32]
prob_cookies = [0, 0.2, 0.4, 0.3, 0.1]
prob_muffins = [0.2, 0.4, 0.3, 0.1, 0]

plt.figure(figsize=(8, 6))

plt.subplot(2, 1, 1)
plt.bar(ages, prob_cookies, color='blue', width=2, alpha=0.7)
plt.title('P(age | cookies)')
plt.xlabel('Age')
plt.ylabel('Probability')
plt.xticks(ages)
plt.ylim(0, 0.5)

plt.subplot(2, 1, 2)
plt.bar(ages, prob_muffins, color='orange', width=2, alpha=0.7)
plt.title('P(age | muffins)')
plt.xlabel('Age')
plt.ylabel('Probability')
plt.xticks(ages)
plt.ylim(0, 0.5)

plt.tight_layout()
plt.show()
```

✓ 0.0s



1a) Andrew Onozuka A16760043

b)

A good prior in this example for *P(cookies)* and *P(muffins)* is ⅓ and ⅔ respectively, as the amount of total cookies is 100 to 200 for muffins, over the total of 300.

work:
Cookies: 0+20+40+30+10=100
Muffins: 40+80+60+20+0=200
Total: 100 + 200 = 300

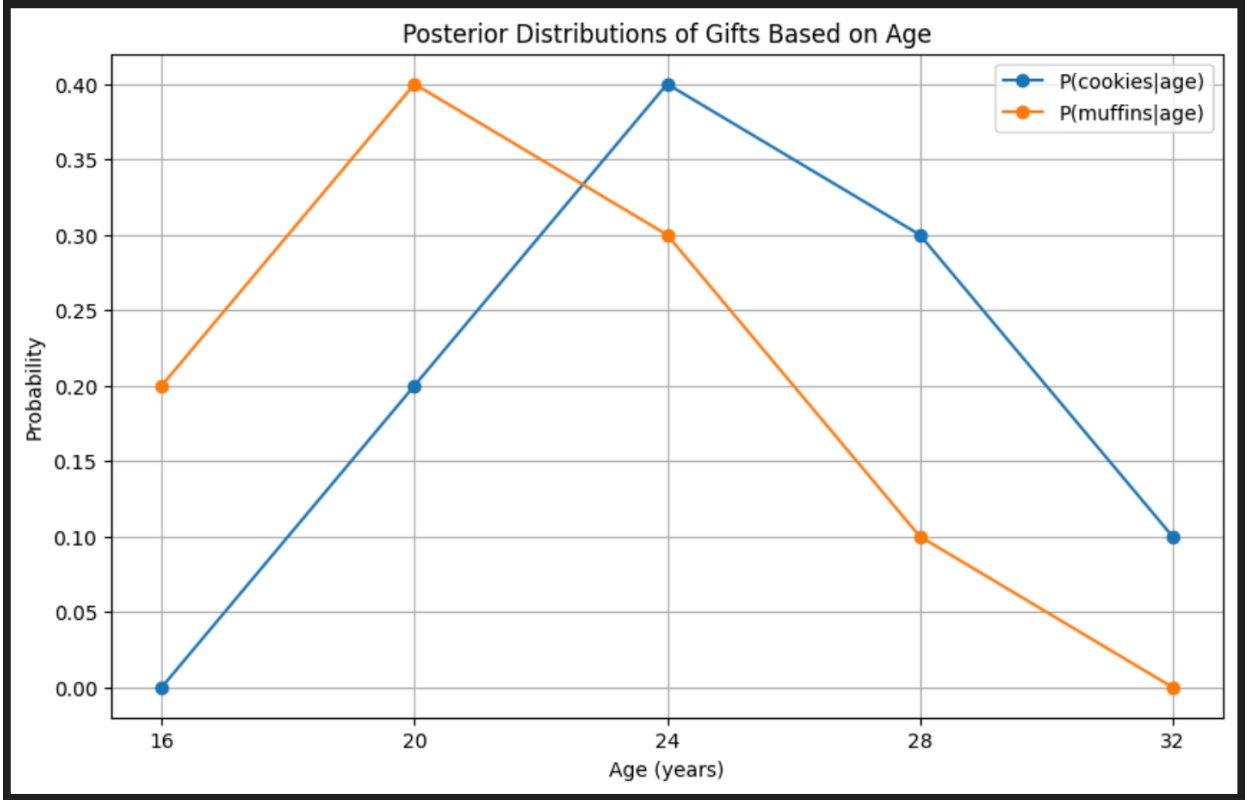
1a) Andrew Onozuka A16760043

c)

```python
import numpy as np
import matplotlib.pyplot as plt

cookies_counts = np.array([0, 20, 40, 30, 10])
muffins_counts = np.array([40, 80, 60, 20, 0])
age_bins = np.array([16, 20, 24, 28, 32])

total_cookies = np.sum(
total_muffins = np.sum(muffins_counts)

# Prior probabilities
prior_cookies = total_cookies / (total_cookies + total_muffins)
prior_muffins = total_muffins / (total_cookies + total_muffins)

# Posterior distributions
posterior_cookies = cookies_counts / total_cookies
posterior_muffins = muffins_counts / total_muffins

plt.figure(figsize=(10, 6))

plt.plot(age_bins, posterior_cookies, marker='o', linestyle='-', label='P(cookies|age)')
plt.plot(age_bins, posterior_muffins, marker='o', linestyle='-', label='P(muffins|age)')

plt.title('Posterior Distributions of Gifts Based on Age')
plt.xlabel('Age (years)')
plt.ylabel('Probability')
plt.xticks(age_bins)
plt.legend()
plt.grid(True)
plt.show()
```

(variable) cookies_counts: NDArray[Any]

✓ 0.0s



1a) Andrew Onozuka A16760043

d)

Age 31: Cookie
Age 21: Muffin
Age 23: Cookie

```python
import numpy as np
import matplotlib.pyplot as plt

ages = np.array([16, 20, 24, 28, 32])

likelihood_cookies = np.array([0, 20, 40, 30, 10])
likelihood_muffins = np.array([40, 80, 60, 20, 0])

prior_cookies = 1/3
prior_muffins = 2/3

posterior_cookies = likelihood_cookies * prior_cookies
posterior_muffins = likelihood_muffins * prior_muffins

posterior_cookies /= np.sum(posterior_cookies)
posterior_muffins /= np.sum(posterior_muffins)

# MAP
guest_ages = [31, 21, 23]
for age in guest_ages:
    idx = np.argmin(np.abs(ages - age))
    prob_cookie_given_age = posterior_cookies[idx]
    prob_muffin_given_age = posterior_muffins[idx]

    print(f"Age {age}:")
    print(f"P|Cookie: {prob_cookie_given_age * 100:.2f}%")
    print(f"P|Muffin: {prob_muffin_given_age * 100:.2f}%")

    if prob_cookie_given_age > prob_muffin_given_age:
        print("MAP Estimate: Cookie")
    else:
        print("MAP Estimate: Muffin")
    print()
```

✓ 0.0s

```
Age 31:
P|Cookie: 10.00%
P|Muffin: 0.00%
MAP Estimate: Cookie

Age 21:
P|Cookie: 20.00%
P|Muffin: 40.00%
MAP Estimate: Muffin

Age 23:
P|Cookie: 40.00%
P|Muffin: 30.00%
MAP Estimate: Cookie
```

1a) Andrew Onozuka A16760043

Problem 3

a)
The matrix F with my PID cannot be a fundamental matrix. The determinant is not zero.

b)

```python
import numpy as np

def estimate_fundamental_matrix(correspondences):
    A = np.zeros((len(correspondences), 9))
    for i, (x, x_prime) in enumerate(correspondences):
        A[i] = [x[0]*x_prime[0], x[0]*x_prime[1], x[0], x[1]*x_prime[0], x[1]*x_prime[1], x[1], x_prime[0], x_prime[1], 1]

    _, _, V = np.linalg.svd(A)
    F = V[-1].reshape(3, 3)

    U, S, V = np.linalg.svd(F)
    S[-1] = 0
    F = np.dot(U, np.dot(np.diag(S), V))

    return F

correspondences = [((x1, y1), (x1_prime, y1_prime)), ((x2, y2), (x2_prime, y2_prime)), ...]

F = estimate_fundamental_matrix(correspondences)
print("Estimated Fundamental Matrix:\n", F)
```

c)

```python
U, S, V = np.linalg.svd(F_prime)

g = V[-1]
h = U[-1]

print("Vector g:\n", g)
print("Vector h:\n", h)
```

d)
ran out of time

1a) Andrew Onozuka A16760043

Problem 4

a)

```python
import numpy as np

def sigmoid(z):
    return 1 / (1 + np.exp(-z))

def sigmoid_derivative(z):
    return sigmoid(z) * (1 - sigmoid(z))

# L4 regularization term
def L4_regularization(w):
    return np.sum(np.abs(w)**4)

# unregularized cost function
def E0(y, a):
    return 0.5 * (y - a)**2

# partial derivatives of E0 with respect to w1, w2, and w3
def partial_E0(y, a, x, w):
    return (y - a) * sigmoid_derivative(np.dot(w, x))

# partial derivatives of L4 with respect to w1, w2, and w3
def partial_L4(w):
    return 4 * np.abs(w)**3 * np.sign(w)

# update rule for weights using gradient descent with L4 regularization
def update_weights(y, a, x, w, learning_rate):
    dE0_dw = partial_E0(y, a, x, w)
    dL4_dw = partial_L4(w)
    regularization_term = dE0_dw + dL4_dw
    return w - learning_rate * regularization_term
```

1a) Andrew Onozuka A16760043

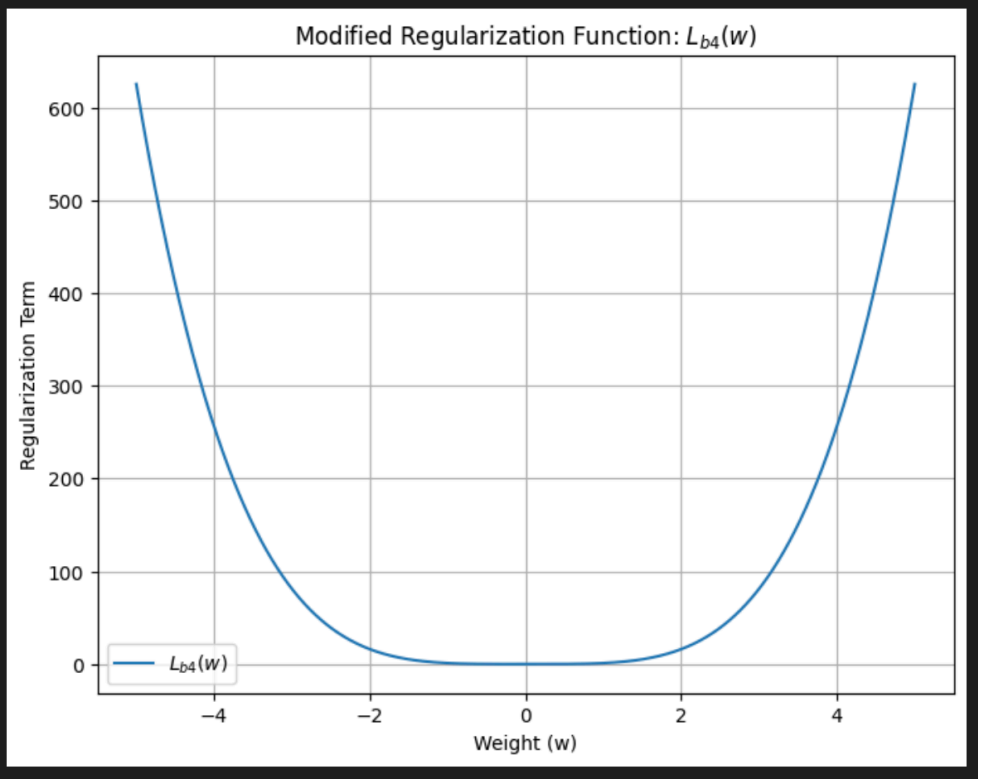
b)

One potential drawback of using L_4 regularization is that with a long-tailed distribution of weights, it may not effectively penalize larger weights compared to an L_2 regularization. Because L_4 regularization computes the penalties based on the fourth power of the weights, it may not influence the large weights as effectively as L_2 regularization, meaning it may be overfitted as the large weights dominated the optimization process. Because L_2 regularization is on the squared magnitude of the weights, it usually has a stronger effect on larger weights. As a result, for the case of a long-tailed distribution of weights, L_2 regularization is likely more effective than L_4 regularization.

c)

```python
import numpy as np
import matplotlib.pyplot as plt

def L4c(w, c):
    return np.where(np.abs(w) < c, w**4, c**4)

w_range = np.linspace(-5, 5, 1000)  # sample range
c = max(6, 1) # replace alpha2 with 6 for A16760043
L4c_values = L4c(w_range, c)

plt.figure(figsize=(8, 6))
plt.plot(w_range, L4c_values, label=r'$L_{b4}(w)$')
plt.xlabel('Weight (w)')
plt.ylabel('Regularization Term')
plt.title('Modified Regularization Function: $L_{b4}(w)$')
plt.grid(True)
plt.legend()
plt.show()
```

✓ 0.0s



1a) Andrew Onozuka A16760043

d)
The advantage of the modified regularization function $\hat{L}_4$ over L_4 in the context of optimizing the cost function E, especially with a long-tailed initial distribution of weights, lies in its ability to provide stronger regularization for extreme weight values. While traditional L_4 regularization may not effectively penalize larger weights compared to L_2 regularization due to its computation based on the fourth power of the weights, $\hat{L}_4$ addresses this issue by applying a stronger penalty to weights with extreme values. By increasing the regularization term significantly for weights exceeding a threshold determined based on the maximum of 6 and 1, $\hat{L}_4$ ensures better control over the influence of outliers and extreme weights during optimization, leading to improved convergence.

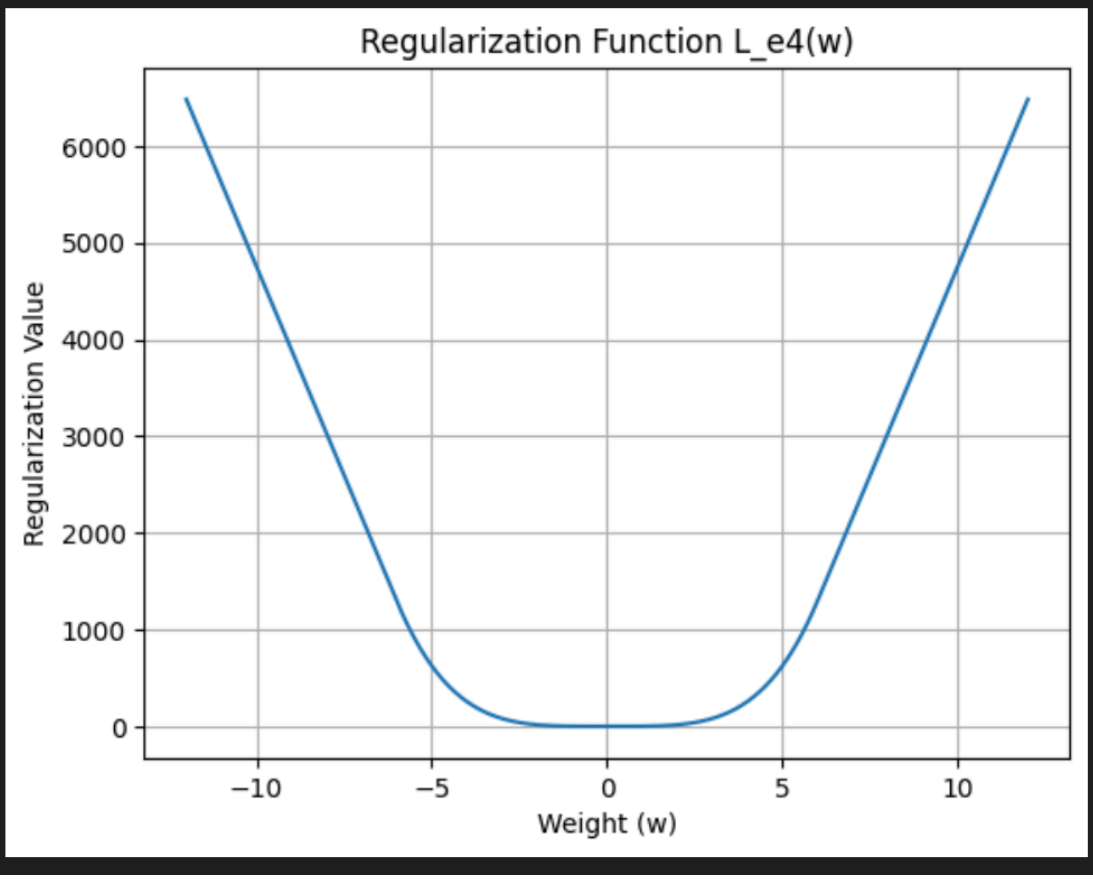
e)
One potential drawback of using the modified regularization function $\hat{L}_4$ in part (c) compared to L_4 in part (b) for optimizing the cost function E is the increased complexity and computational overhead associated with the piecewise definition of the regularization function. While L_4 regularization provides a straightforward penalty based on the fourth power of the weights, $\hat{L}_4$ introduces a conditional expression that requires evaluating the magnitude of each weight and determining whether it exceeds a certain threshold. This additional computational burden can impact training efficiency, especially when dealing with large-scale datasets or complex neural network architectures. Additionally, the introduction of a threshold parameter adds another hyperparameter to tune, which may require additional optimization efforts and could potentially lead to suboptimal regularization behavior if not properly chosen.

1a) Andrew Onozuka A16760043

f)

```python
import numpy as np
import matplotlib.pyplot as plt

def L4f(w, c):
    if np.abs(w) < c:
        return w**4
    else:
        return 4 * c**3 * np.abs(w) - 3 * c**4

w_range = np.linspace(-2*c, 2*c, 1000)
c = max(6, 1)
L4f_values = [L4f(w, c) for w in w_range]

# Plot the regularization function
plt.plot(w_range, L4f_values)
plt.xlabel('Weight (w)')
plt.ylabel('Regularization Value')
plt.title('Regularization Function L_e4(w)')
plt.grid(True)
plt.show()
```

✓ 0.0s



1a) Andrew Onozuka A16760043

g)
i) We first overcame the drawback of regularization L_4 in part b by providing a smoother transition for the penalties as the weights near the threshold c. Unlike L_4 which switches abruptly from w^4 to c^4 at |w| = c, we now smoothly transition between penalty terms, leading to a more gradual regularization effect as we approach c.

ii) The new regularization function in part f is more suitable for our optimization than the part in c because it provides a more flexible penalty scheme that can adapt to the magnitude of the weights, while previously it was fixed, regardless of the magnitude. This adaptive regularization ensures that larger weights are more effectively regularized, which allows for better generalization performance than before.

Problem 5

(a)

$$u = \frac{fP}{Q}$$

$$u' = \frac{f(P-b)}{Q}$$

(b)

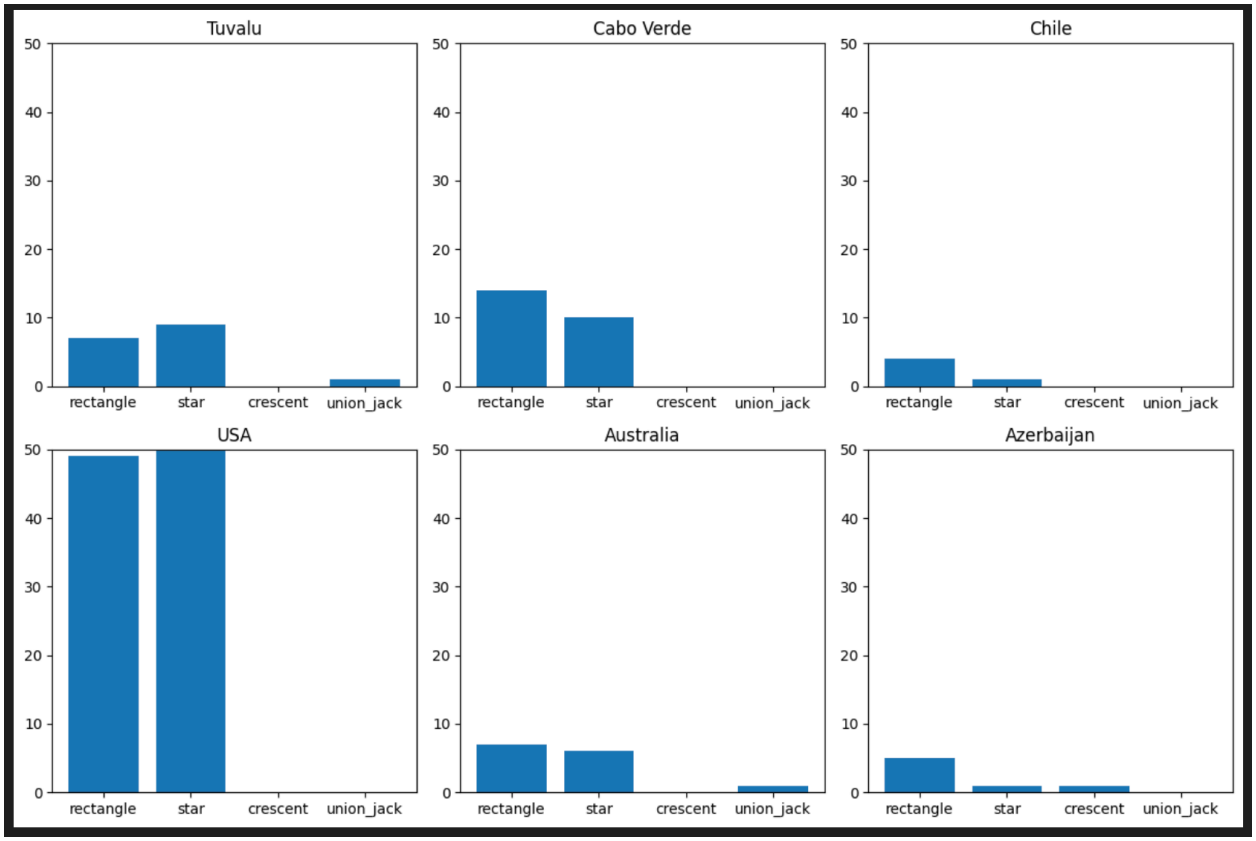
$$Q = \frac{fP}{u - u'}$$

1a) Andrew Onozuka A16760043

Problem 6

a)

```python
import matplotlib.pyplot as plt

def plot_histograms(countries, histograms):
    fig, axs = plt.subplots(2, 3, figsize=(12, 8))
    max_y = max(max(h.values()) for h in histograms)
    for i in range(len(countries)):
        row = i // 3
        col = i % 3
        ax = axs[row, col]
        ax.bar(histograms[i].keys(), histograms[i].values())
        ax.set_title(countries[i])
        ax.set_ylim(0, max_y)  # Set the same y-axis range for all subplots
    plt.tight_layout()
    plt.show()

# Example usage:
countries = ['Tuvalu', 'Cabo Verde', 'Chile', 'USA', 'Australia', 'Azerbaijan']
histograms = [
    {'rectangle': (1 + 4 + 2), 'star': 9, 'crescent': 0, 'union_jack': 1},  # Tuvalu - rectangles in top left, 4 between the borders of the union jack against the red, and then 2 for the red cross
    {'rectangle': (5 + 4 + 3 + 2), 'star': 10, 'crescent': 0, 'union_jack': 0},  # Cabo Verde - not sure if we are counting overlapping rectangles as directions say features are robust to color variations? would be +4+3+2 if we were to merge 2, 3, and 4 rectangles into 1
    {'rectangle': (3 + 1), 'star': 1, 'crescent': 0, 'union_jack': 0},  # Chile - blue and white merged is one rectangle
    {'rectangle': (1 + 13 + 6 + 5 + 4 + 3 + 2 + 1 + 5 + 4 + 3 + 2), 'star': 50, 'crescent': 0, 'union_jack': 0},  # USA - top left + 13 rectangles + top 7 merged 6 + 5 + 4 + 3 + 2 (2, 3, 4, 5, 6) + big top 1 + bottom merged 5 + 4 + 3 + 2 (2, 3, 4, 5)
    {'rectangle': (1 + 4 + 2), 'star': 6, 'crescent': 0, 'union_jack': 1},  # Australia
    {'rectangle': (3 + 2), 'star': 1, 'crescent': 1, 'union_jack': 0}  # Azerbaijan
]

plot_histograms(countries, histograms)
```



* calculated rectangles with the possibility of joint rectangles, minus the largest one outlined by the border of the flag. this is why USA has many triangles, as the top 7 rectangles and the bottom 6 rectangles can be merged to be one rectangle made up of up to 6 rectangles.

Tuvalu - rectangles in top left, 4 between the borders of the union jack against the red, and then 2 for the red cross
Cabo Verde - not sure if we are counting overlapping rectangles as directions say features are robust to color variations? would be +4+3+2 if we were to merge 2, 3, and 4 rectangles into 1
Chile - blue and white merged is one rectangle
USA - top left + 13 rectangles + top 7 merged 6 + 5 + 4 + 3 + 2 (2, 3, 4, 5, 6) + big top 1 + bottom merged 5 + 4 + 3 + 2 (2, 3, 4, 5)

1a) Andrew Onozuka A16760043

b)

```python
import numpy as np

def cosine_similarity(vector1, vector2):
    dot_product = np.dot(vector1, vector2)
    norm_vector1 = np.linalg.norm(vector1)
    norm_vector2 = np.linalg.norm(vector2)
    similarity = dot_product / (norm_vector1 * norm_vector2)
    return similarity

tuvalu_features = np.array([1 + 4 + 2, 9, 0, 1])  # Tuvalu features: rectangle, star, crescent, union_jack
candidate_features = [
    np.array([1 + 4 + 2, 9, 0, 1]),  # Cabo Verde
    np.array([5 + 4 + 3 + 2, 10, 0, 0]),  # Chile
    np.array([3 + 1, 1, 0, 0]),  # USA
    np.array([1 + 4 + 2, 6, 0, 1]),  # Australia
    np.array([3 + 2, 1, 1, 0])  # Azerbaijan
]

similarities = []
for features in candidate_features:
    similarity = cosine_similarity(tuvalu_features, features)
    similarities.append(similarity)

for i, similarity in enumerate(similarities):
    print(f"Similarity between Tuvalu and {countries[i+1]}: {similarity}")
```

✓ 0.0s

```
Similarity between Tuvalu and Cabo Verde: 1.0
Similarity between Tuvalu and Chile: 0.9547207153487725
Similarity between Tuvalu and USA: 0.7840461300725386
Similarity between Tuvalu and Australia: 0.9798248864440795
Similarity between Tuvalu and Azerbaijan: 0.7398354660478001
```

c)

We choose Australia and Chile as our first and second most preferred trade partners, but this is assuming that by flag similarity that the countries would be good to trade with. In reality, we should probably look at what our country can offer in terms of exports, and see which countries we could both benefit from trading with.

1a) Andrew Onozuka A16760043

Problem 7

a)
The thing about the pattern of behavior of the histogram that stood out to me the most is the red line remaining almost vertical until the layer 13. Given that the histograms represent the distributions of the derivatives after training, this means that the layers in which the red curve is a vertical line have very low variance and are highly concentrated around zero.

b)
This behavior is concerning and poses a significant challenge to learning, because the late convergence means that the model is not learning to make accurate predictions until very late in the CNN. With the histograms remaining almost vertical up to layer 13, this suggests that the activations in the earlier layers might be experiencing a saturation effect, where the activation function squeezes the inputs into too narrow of a range, resulting in gradients close to zero (vanishing gradients). This limits the ability of the model to learn effectively and this saturation effect can pose significant problems by impeding the flow of gradient information during backpropagation early on in the training.

c)
We can modify the network architecture by adding additional loss layers at previous layers. I'm not sure by what is desired by draw a figure, but because we can see that the 2nd loss layer helped tremendously, we can try to put it earlier in the network rather than towards the end.

d)
The behavior depicted by the histograms in black, where they exhibit a broader spread earlier in the network compared to the red histograms, suggests that applying loss at multiple stages of the network has influenced the activations differently. By imposing loss at intermediate layers, the network is encouraged to learn more discriminative features earlier in the training process, leading to a more distributed representation of derivatives across the neurons.

e)
In class, we discussed the concept of residual learning, which forms the basis of Residual Networks (ResNets). Residual learning aims to ease the training of deep convolutional neural networks (CNNs) by introducing skip connections, also known as shortcut connections, that allow the network to learn residual mappings. These skip connections enable the gradient to flow more directly through the network during backpropagation, mitigating the vanishing gradient problem that often hinders the training of very deep networks. By allowing the network to learn the residual functions, rather than directly learning the desired underlying mapping, ResNets facilitate the training of much deeper architectures, resulting in improved performance and convergence.

Problem 8
a) no comments
b) no comments