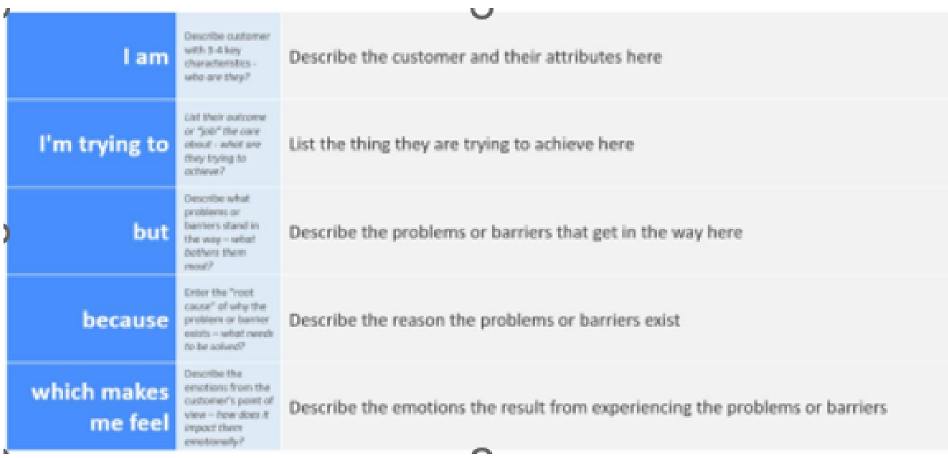
Final Project Report Template

Project Initialization and Planning Phase

Date	15 March 2024
Team ID	SWTID1720439521
Project Name	Covidvision: Advanced Covid-19 Detection
Maximum Marks	3 Marks

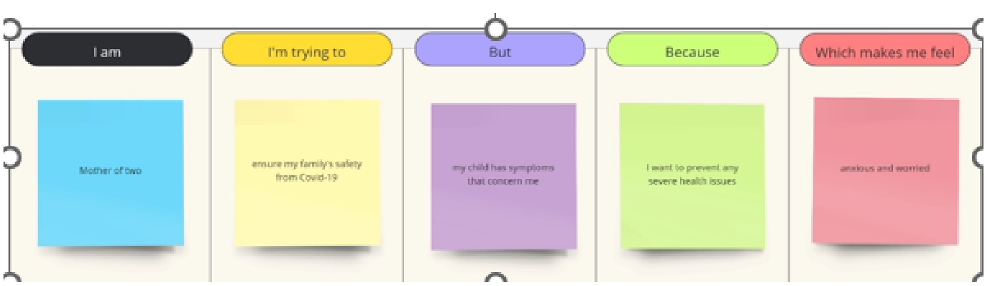
Define Problem Statements (Customer Problem Statement Template):

Medical professionals and patients face significant challenges in the timely and accurate detection of Covid-19, leading to delayed treatment and increased transmission rates. Covidvision aims to address these issues by leveraging advanced deep learning techniques to analyze lung X-rays, providing rapid and precise Covid-19 detection. This innovative solution seeks to enhance diagnostic accuracy, reduce the burden on healthcare systems, and ultimately improve patient outcomes during the pandemic.



Reference: https://miro.com/templates/customer-problem-statement/

Example:



Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	elderly person	understand my own Covid-19 diagnosis	it's difficult to get clear information from doctors	I have underlying health conditions	frustrated and scared
PS-2	Working Employee	Get a fast diagnosis to clear my suspicion of covid	It takes minimum 2 days to get a diagnosis done	Of the slow traditional process of testing	stressed and overwhelmed

Project Initialization and Planning Phase

Date	15 March 2024
Team ID	SWTlD1720439521
Project Title	Covidvision: Advanced Covid-19 Detection From Lung X-Rays With Deep Learning
Maximum Marks	3 Marks

Project Proposal (Proposed Solution) template

"Covid-19 Detection from Lung X-rays" utilizes deep learning algorithms to analyze lung X-ray images for signs of Covid-19 infection. By leveraging vast datasets and image recognition technology, this project aims to provide accurate and rapid diagnosis, aiding in early detection and containment of the virus.

Project Overview	
Objective	The main objective of this project is to detect the presence of Covid-19 in a person using his chest X-rays by implementing Deep Learning techniques and integrating AI into the field of medicine
Scope	The project efficiently detects the presence of covid-19 in a person, therefore in places with overwhelming cases, or in places with no access to expert radiologists, this model can be very useful. It also has the ability to detect signs of viral pneumonia.
Problem Statement	
Description	Addressing the delay in the result and chance of error in the current system to test covid-19, this project stands to triumph over the present system by efficiently and almost immediately producing the test results.
Impact	Every second is crucial in the field of medicine, it might even cost a life. This project was developed to tackle one such problem. This gives you a very accurate test result in a span of seconds which usually took over a day. This project totally revolutionizes the process of covid detection

Proposed Solution	
Approach	To use Deep Learning to train a model with a dataset consisting of covid, normal and viral pneumonia x-ray images.
Key Features	Implementation of ANN-based detection model

Resource Requirements

Resource Type	**Description**	**Specification/Allocation**
Hardware		
Computing Resources	CPU/GPU specifications, number of cores	Intel core i7, Nvidia RTX4060
Memory	RAM specifications	16 GB
Storage	Disk space for data, models, and logs	1 TB SSD
Software		
Frameworks	Python frameworks	Flask
Libraries	Additional libraries	Tensorflow, pandas, scikit-learn, mathplotlib
Development Environment	IDE, version control	Jupyter Notebook, Git
Data		
Data	Source, size, format	Acquired dataset, 1000 images

Initial Project Planning Template

Date	15 March 2024
Team ID	SWTID1720439521
Project Name	Covidvision: Advanced Covid-19 Detection From Lung X-Rays With Deep Learning
Maximum Marks	4 Marks

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Functional	**User**	**User Story / Task**	**Priority**	**Team**	**Sprint**	**Sprint End**
Data collection	USN-1	Finding suitable dataset	High	Lohith, Santript	2024-06-28	2024-06-30
Data preprocessing	USN-2	Data Standardization, Augmentation, and Data splitting	High	Santript	2024-06-30	2024-07-4
Model Training	USN-3	Training and evaluating the model	high	Santript	2024-07-05	2024-07-18
Web-Page Building	USN-4	Creating the front-end i.e. HTML and CSS	Medium	Pratham, Somya	2024-07-07	2024-07-15
Backend	USN-5	Building and integrating backend	High	Santript, Lohith	2024-07-15	2024-07-17
Templates	USN-6	Project details record	Medium	Somya, Lohith	2024-06-30	2024-07-17
Web integrations and deployment		Uploading all the files and deploying	High	Lohith, Santript, Somya, Pratham	2024-07-20	2024-07-21

Data Collection and Preprocessing Phase

Date	15 March 2024
Team ID	SWTlD1720439521
Project Title	Covid vision: Advanced Covid-19 Detection From Lung X-Rays With Deep Learning
Maximum Marks	2 Marks

Data Collection Plan & Raw Data Sources Identification Template

Elevate your data strategy with the Data Collection plan and the Raw Data Sources report, ensuring meticulous data curation and integrity for informed decision-making in every analysis and decision-making endeavor.

Data Collection Plan Template

Section	Description
Project Overview	This Deep Learning model aims to speed up and accurate the process covid diagnosis. Using a dataset that contains X-ray images of normal, people suffering with covid and people suffering with pneumonia, this model seamlessly detects and diagnosis the person using his chest X-ray.
Data Collection Plan	Search for datasets containing the X-ray images of the mentioned types.
Raw Data Sources Identified	The raw data collection sources of this model involves kaggle and youtube

Raw Data Sources Template

Source Name	Description	Location/URL	Format	Size	Access Permissions
kaggle	This dataset consists of 3 folders containing	https://www.kaggle.com/code/rollanmaratov/covid19-detection-using-tensorflow-from-chest-xray/data	image	3 GB	Public
Youtube	This dataset is pre-split into test and train. Therefore	https://drive.google.com/drive/folders/1Ax8PttO8sC2_uWFwqo570lz-4Q3gCsu9	Image	1 GB	public

Data Collection and Preprocessing Phase

Date	15 March 2024
Team ID	SWTlD1720439521
Project Title	Covid vision: Advanced Covid-19 Detection From Lung X-Rays With Deep Learning
Maximum Marks	2 Marks

Data Quality Report Template

The Data Quality Report Template will summarize data quality issues from the selected source, including severity levels and resolution plans. It will aid in systematically identifying and rectifying data discrepancies.

Data Source	Data Quality Issue	Severity	Resolution Plan
Dataset	Contained unnecessary data not useful for the model being built	Low	The noise in the data was removed from the dataset and only the appropriate data was used

Data Collection and Preprocessing Phase

Date	15 March 2024
Team ID	SWTID1720439521
Project Title	Covidvision: Advanced Covid-19 Detection From Lung X-Rays With Deep Learning
Maximum Marks	6 Marks

Preprocessing Template

The images will be preprocessed by resizing, normalizing, augmenting, denoising, adjusting contrast, detecting edges, converting color space, cropping, batch normalizing, and whitening data. These steps will enhance data quality, promote model generalization, and improve convergence during neural network training, ensuring robust and efficient performance across various computer vision tasks.

Section	**Description**
Data Overview	This dataset is divided in the ration 75:25 and contains the images of Covid positive, normal and viral pneumonia.
Resizing	Resize images to a specified target size.
Normalization	Normalize pixel values to a specific range.
Data Augmentation	Apply augmentation techniques such as flipping, rotation, shifting, zooming, or shearing.
Image Cropping	Crop images to focus on the regions containing objects of interest.
Data Preprocessing Code Screenshots	

Loading Data	`data_path_train=r"C:\Users\lohit\OneDrive\Desktop\project\team_dataset\train"` `data_path_test=r"C:\Users\lohit\OneDrive\Desktop\project\team_dataset\test"`

Resizing	`img_size=120`
	`img_transform=transforms.Compose([transforms.Resize((img_size,img_size)),`
	`transforms.RandomHorizontalFlip(),`
	`transforms.ToTensor(),`
	`transforms.Normalize(mean=[0.485,0.456,0.475],std=[0.229,0.224,0.225])])`

Normalization	`img_size=120`
	`img_transform=transforms.Compose([transforms.Resize((img_size,img_size)),`
	`transforms.RandomHorizontalFlip(),`
	`transforms.ToTensor(),`
	`transforms.Normalize(mean=[0.485,0.456,0.475],std=[0.229,0.224,0.225])])`

Data Augmentation	`img_size=120`
	`img_transform=transforms.Compose([transforms.Resize((img_size,img_size)),`
	`transforms.RandomHorizontalFlip(),`
	`transforms.ToTensor(),`
	`transforms.Normalize(mean=[0.485,0.456,0.475],std=[0.229,0.224,0.225])])`

\Image Cropping	`for img,label in train_loader:`
	`print(img.shape)`
	`break`

Model Development Phase Template

Date	15 March 2024
Team ID	SWTID1720439521
Project Title	Covidvision: Advanced Covid-19 Detection From Lung X-Rays With Deep Learning
Maximum Marks	5 Marks

Model Selection Report

In the model selection report for future deep learning and computer vision projects, various architectures, such as CNNs or RNNs, will be evaluated. Factors such as performance, complexity, and computational requirements will be considered to determine the most suitable model for the task at hand.

Model Selection Report:

Model	**Description**
Model 1	**Convolutional Neural Network (CNN)**: Convolutional layers are used by convolutional neural networks, or CNNs, to automatically and adaptively learn the spatial hierarchies of features. CNNs are specifically made for picture data. For tasks involving segmentation, object detection, and picture classification, they are very successful. **Accuracy: 85%**
Model 2	**Artificial Neural Networks (ANNs):** ANNs are made up of several layers of connected neurons, each of which has a weight assigned to it. They are adaptable and suitable for many applications, including as regression and classification. For jobs involving images, they could not be as effective as specialist designs like CNNs. **Accuracy: 90%**

Evaluation Standards:

Performance: Recall, accuracy, and precision.

Complexity: The quantity of parameters and network depth.

Computational requirements: Memory utilization, inference time, and training time are all considered computational requirements.

Model Development Phase Template

Date	15 March 2024
Team ID	SWTID1720439521
Project Title	Covidvision: Advanced Covid-19 Detection From Lung X-Rays With Deep Learning
Maximum Marks	10 Marks

Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include a summary and training and validation performance metrics for multiple models, presented through respective screenshots.

Initial Model Training Code (5 marks):

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader
from torch.utils.data import random_split

import  torchvision
import torchvision.transforms as transforms
from torchvision.datasets import ImageFolder
from torchvision.utils import make_grid
```

```
data_path_train=r"C:\Users\lohit\OneDrive\Desktop\project\team_dataset\train"
```

```
data_path_test=r"C:\Users\lohit\OneDrive\Desktop\project\team_dataset\test"
```

```
img_size=120
img_transform=transforms.Compose([transforms.Resize((img_size,img_size)),
                                  transforms.RandomHorizontalFlip(),
                                  transforms.ToTensor(),
                                  transforms.Normalize(mean=[0.485,0.456,0.475],std=[0.229,0.224,0.225])])
```

```
train_data=ImageFolder(root=data_path_train,transform=img_transform)
test_data=ImageFolder(root=data_path_test,transform=img_transform)
```

```
len(train_data),len(test_data)
```

```
train_data.class_to_idx
```

```
val_data,test_data=random_split(test_data,[50,16])
```

```
len(val_data),len(test_data)
```

```
train_loader=DataLoader(train_data,batch_size=50,shuffle=True)
val_loader=DataLoader(val_data,batch_size=50,shuffle=True)
```

```
for img,label in train_loader:
    print(img.shape)
    break
```

```
def show_img(data):
    for img,label in data:
        plt.figure(figsize=(10,10))
        plt.imshow(make_grid(img,nrow=5).permute(1,2,0))
        plt.show()
        break
```

```
show_img(train_loader)
```

```
show_img(val_loader)
```

```python
class ANN(nn.Module):
    def __init__(self,hidden_layer=64):
        super(ANN,self).__init__()
        self.fc1=nn.Linear(120*120*3,hidden_layer)
        self.fc2=nn.Linear(hidden_layer,3)
        self.relu=nn.ReLU()
    def forward(self,img):
        out=img.view(-1,120*120*3)
        out=self.fc1(out)
        out=self.relu(out)
        out=self.fc2(out)
        return out
```

```python
import matplotlib.pyplot as plt

def train(model,  loss_fn, optimizer):
    epochs=15
    training_loss = []
    training_acc = []
    validation_loss = []
    validation_acc = []

    for epoch in range(epochs):
        train_loss = 0.0
        train_acc = 0.0
        model.train()

        # Training Loop
        for images, labels in train_loader:
            optimizer.zero_grad()
            output = model(images)
            loss = loss_fn(output, labels)
            loss.backward()
            optimizer.step()

            predictions = torch.argmax(output, 1)
            train_acc += (predictions == labels).sum().item()
            train_loss += loss.item()

        training_acc.append(train_acc / len(train_loader.dataset))
        training_loss.append(train_loss / len(train_loader))

        # Validation Loop
        val_loss = 0.0
        val_acc = 0.0
        model.eval()
```

```python
    with torch.no_grad():
        for images, labels in val_loader:
            output = model(images)
            loss = loss_fn(output, labels)

            predictions = torch.argmax(output, 1)
            val_acc += (predictions == labels).sum().item()
            val_loss += loss.item()

    validation_acc.append(val_acc / len(val_loader.dataset))
    validation_loss.append(val_loss / len(val_loader))

    # Print epoch statistics
    print('Epoch {}, Training Loss: {:.4f}, Training Acc: {:.4f}, Validation Loss: {:.4f}, Validation Acc: {:.4f}'
          .format(epoch + 1, train_loss / len(train_loader), train_acc / len(train_loader.dataset),
                  val_loss / len(val_loader), val_acc / len(val_loader.dataset)))

plt.title('Accuracy vs Epoch')
plt.plot(range(epochs),training_acc,label='training accuracy')
plt.plot(range(epochs),validation_acc,label='validation accuracy')
plt.legend()
plt.xlabel('Epochs')
plt.ylabel('Training\Validation Accuracy')
plt.show()
```

```python
# Assuming train_loader, val_loader, loss_fn, optimizer are defined

# Train the model
train(model,loss_fn, optimizer)
```

```python
def predict_img(img,model):
    x=img.unsqueeze(0)
    y=model(x)
    pred=torch.argmax(y,dim=1)
    return train_data.classes[pred]
```

```python
import torch
import matplotlib.pyplot as plt

# Assuming test_data, train_data, and predict_img are defined

# Function to make a prediction on a single image
def predict_img(img, model):
    model.eval()
    with torch.no_grad():
        img = img.unsqueeze(0)  # Add batch dimension
        output = model(img)
        prediction = torch.argmax(output, 1)
    return prediction.item()

# Get an image and its label from the test dataset
img, label = test_data[2]

# Display the image
plt.imshow(img.permute(1, 2, 0))
plt.title(f'Actual Label: {train_data.classes[label]}')
plt.show()

# Predict the label for the image using the model
predicted_label = predict_img(img, model)
print('Actual Label:', train_data.classes[label], 'Prediction label:', train_data.classes[predicted_label])
```

```python
img, label = test_data[10]

# Display the image
plt.imshow(img.permute(1, 2, 0))
plt.title(f'Actual Label: {train_data.classes[label]}')
plt.show()

# Predict the label for the image using the model
predicted_label = predict_img(img, model)
print('Actual Label:', train_data.classes[label], 'Prediction label:', train_data.classes[predicted_label])
```

```python
img, label = test_data[15]

# Display the image
plt.imshow(img.permute(1, 2, 0))
plt.title(f'Actual Label: {train_data.classes[label]}')
plt.show()

# Predict the label for the image using the model
predicted_label = predict_img(img, model)
print('Actual Label:', train_data.classes[label], 'Prediction label:', train_data.classes[predicted_label])
```

```
img, label = test_data[14]

# Display the image
plt.imshow(img.permute(1, 2, 0))
plt.title(f'Actual Label: {train_data.classes[label]}')
plt.show()

# Predict the label for the image using the model
predicted_label = predict_img(img, model)
print('Actual Label:', train_data.classes[label], 'Prediction label:', train_data.classes[predicted_label])
```

```
len(test_data)
```

```
img, label = val_data[40]

# Display the image
plt.imshow(img.permute(1, 2, 0))
plt.title(f'Actual Label: {train_data.classes[label]}')
plt.show()

# Predict the label for the image using the model
predicted_label = predict_img(img, model)
print('Actual Label:', train_data.classes[label], 'Prediction label:', train_data.classes[predicted_label])
```

```
import torch

# Assuming your model is called 'model'
torch.save(model.state_dict(), 'project_model.pth')
```

```
pip install h5py
```

```
import h5py

state_dict = model.state_dict()


with h5py.File('project_model.h5', 'w') as f:
    for key, value in state_dict.items():
        f.create_dataset(key, data=value.cpu().numpy())
```

Model Validation and Evaluation Report (5 marks):

<table>
<tr><th>Model</th><th>Summary</th><th>Training and Validation Performance Metrics</th></tr>
<tr><td>Model 1</td><td>

```python
import matplotlib.pyplot as plt

def train(model,  loss_fn, optimizer):
    epochs=15
    training_loss = []
    training_acc = []
    validation_loss = []
    validation_acc = []

    for epoch in range(epochs):
        train_loss = 0.0
        train_acc = 0.0
        model.train()

        # Training Loop
        for images, labels in train_loader:
            optimizer.zero_grad()
            output = model(images)
            loss = loss_fn(output, labels)
            loss.backward()
            optimizer.step()

            predictions = torch.argmax(output, 1)
            train_acc += (predictions == labels).sum().item()
            train_loss += loss.item()

        training_acc.append(train_acc / len(train_loader.dataset))
        training_loss.append(train_loss / len(train_loader))
```

</td><td>

```
Epoch 1, Training Loss: 0.7760, Training Acc: 0.6135, Validation Loss: 1.5132, Validation Acc:
Epoch 2, Training Loss: 0.5417, Training Acc: 0.7928, Validation Loss: 0.7174, Validation Acc:
Epoch 3, Training Loss: 0.3652, Training Acc: 0.8606, Validation Loss: 0.5825, Validation Acc:
Epoch 4, Training Loss: 0.3747, Training Acc: 0.8566, Validation Loss: 0.6591, Validation Acc:
Epoch 5, Training Loss: 0.3990, Training Acc: 0.8486, Validation Loss: 0.6517, Validation Acc:
Epoch 6, Training Loss: 0.3824, Training Acc: 0.8725, Validation Loss: 2.7382, Validation Acc:
Epoch 7, Training Loss: 0.5186, Training Acc: 0.8167, Validation Loss: 0.5404, Validation Acc:
Epoch 8, Training Loss: 0.2699, Training Acc: 0.9084, Validation Loss: 0.6816, Validation Acc:
Epoch 9, Training Loss: 0.2754, Training Acc: 0.8685, Validation Loss: 0.5456, Validation Acc:
Epoch 10, Training Loss: 0.2403, Training Acc: 0.9084, Validation Loss: 0.6111, Validation Acc:
Epoch 11, Training Loss: 0.2678, Training Acc: 0.9044, Validation Loss: 1.2478, Validation Acc:
Epoch 12, Training Loss: 0.2496, Training Acc: 0.8924, Validation Loss: 0.5200, Validation Acc:
Epoch 13, Training Loss: 0.2416, Training Acc: 0.9243, Validation Loss: 0.9556, Validation Acc:
Epoch 14, Training Loss: 0.2448, Training Acc: 0.9044, Validation Loss: 0.4517, Validation Acc:
Epoch 15, Training Loss: 0.1866, Training Acc: 0.9243, Validation Loss: 0.4795, Validation Acc:
```

</td></tr>
</table>

Model Optimization and Tuning Phase Template

Date	15 March 2024
Team ID	SWTID1720439521
Project Title	Covidvision: Advanced Covid-19 Detection From Lung X-Rays With Deep Learning
Maximum Marks	10 Marks

Model Optimization and Tuning Phase

The Model Optimization and Tuning Phase involves refining neural network models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

Hyperparameter Tuning Documentation (8 Marks):

<table>
<tr><th>Model</th><th>Tuned Hyperparameters</th></tr>
<tr><td>ANN</td><td><pre>
class ANN(nn.Module):
    def __init__(self,hidden_layer=64):
        super(ANN,self).__init__()
        self.fc1=nn.Linear(120*120*3,hidden_layer)
        self.fc2=nn.Linear(hidden_layer,3)
        self.relu=nn.ReLU()
    def forward(self,img):
        out=img.view(-1,120*120*3)
        out=self.fc1(out)
        out=self.relu(out)
        out=self.fc2(out)
        return out
</pre></td></tr>
</table>

Final Model Selection Justification (2 Marks):

Final Model	**Reasoning**
Model 1 (or other)	Epoch 1, Training Loss: 0.7760, Training Acc: 0.6135, Validation Loss: 1.5132, Validation Acc: 0.5400 Epoch 2, Training Loss: 0.5417, Training Acc: 0.7928, Validation Loss: 0.7174, Validation Acc: 0.6400 Epoch 3, Training Loss: 0.3652, Training Acc: 0.8606, Validation Loss: 0.5825, Validation Acc: 0.7200 Epoch 4, Training Loss: 0.3747, Training Acc: 0.8566, Validation Loss: 0.6591, Validation Acc: 0.8000 Epoch 5, Training Loss: 0.3990, Training Acc: 0.8486, Validation Loss: 0.6517, Validation Acc: 0.7000 Epoch 6, Training Loss: 0.3824, Training Acc: 0.8725, Validation Loss: 2.7382, Validation Acc: 0.5400 Epoch 7, Training Loss: 0.5186, Training Acc: 0.8167, Validation Loss: 0.5404, Validation Acc: 0.7600 Epoch 8, Training Loss: 0.2699, Training Acc: 0.9084, Validation Loss: 0.6816, Validation Acc: 0.7800 Epoch 9, Training Loss: 0.2754, Training Acc: 0.8685, Validation Loss: 0.5456, Validation Acc: 0.7200 Epoch 10, Training Loss: 0.2403, Training Acc: 0.9084, Validation Loss: 0.6111, Validation Acc: 0.8000 Epoch 11, Training Loss: 0.2678, Training Acc: 0.9044, Validation Loss: 1.2478, Validation Acc: 0.5800 Epoch 12, Training Loss: 0.2496, Training Acc: 0.8924, Validation Loss: 0.5200, Validation Acc: 0.8200 Epoch 13, Training Loss: 0.2416, Training Acc: 0.9243, Validation Loss: 0.9556, Validation Acc: 0.6200 Epoch 14, Training Loss: 0.2448, Training Acc: 0.9044, Validation Loss: 0.4517, Validation Acc: 0.8600 Epoch 15, Training Loss: 0.1866, Training Acc: 0.9243, Validation Loss: 0.4795, Validation Acc: 0.8200