**School of Information Technologies and Engineering, ADA University**

**CSCI4734 – Machine Learning**

**Spring 2024**

**Course Project Report**

# Team 11

|  | **Full Name** | **Student ID** |
| --- | --- | --- |
| **Member 1** | Laman Majid | 11512 |
| **Member 2** | Ulkar Mammadli | 12140 |

# 

# Project

## Problem formulation

In this project, we're diving into the world of cat breed identification using advanced computer vision techniques. Our goal is to develop a system capable of determining whether a cat belongs to one of the five breeds: Domestic Shorthair, Bengal, Ragdoll, Maine Coon, or Siamese. We're leveraging a convolutional neural network (CNN), a powerful type of artificial intelligence renowned for its ability to recognize visual patterns, built with the TensorFlow framework. By training our model on a dataset of images, each labeled with the correct cat breed, the CNN learns to recognize the distinct visual features associated with each breed. The outcome is a sophisticated system that can automatically identify a cat's breed from a photograph, potentially revolutionizing services like pet adoption and customized pet care applications.

## Discussion of related works (optional)

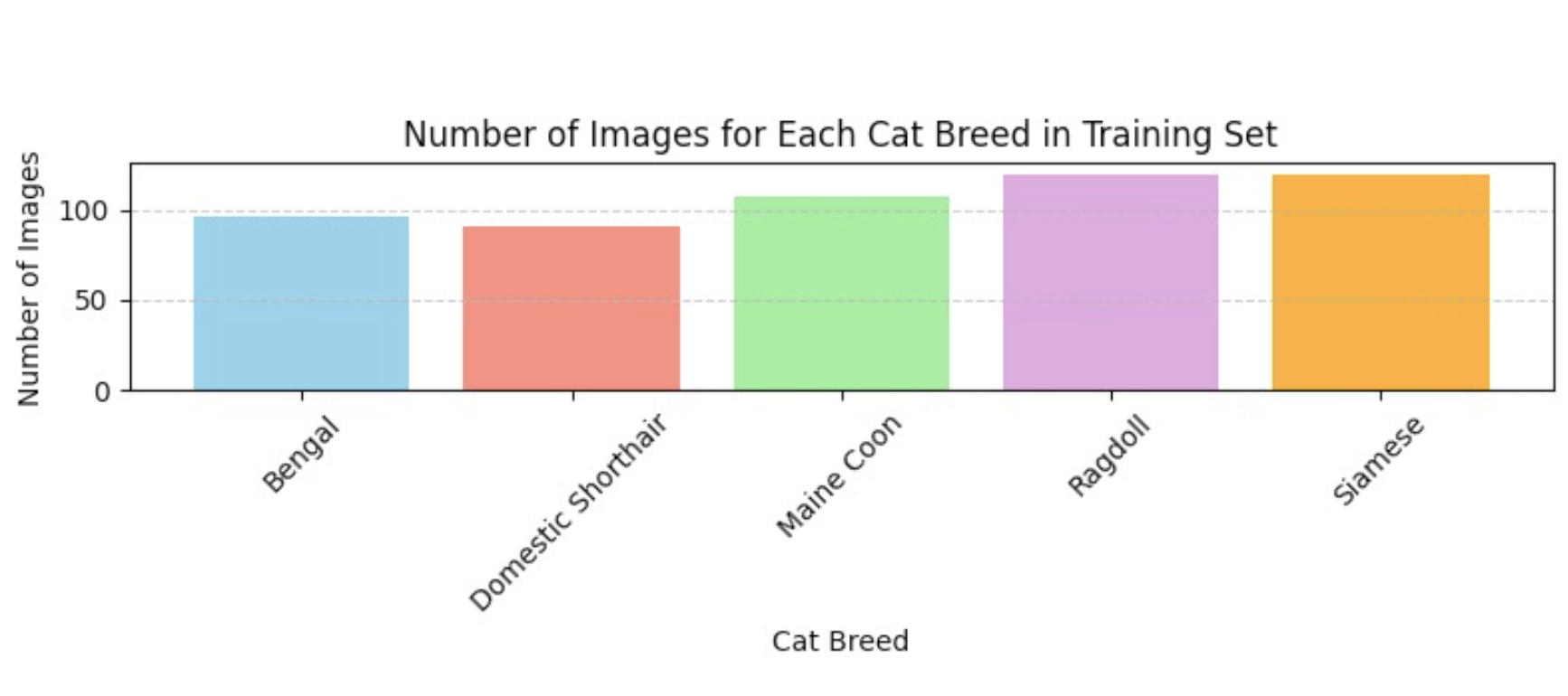
Several studies have explored the potential of image classification across different domains using advanced techniques. One such study focused on Content-Based Image Retrieval (CBIR) for recognizing multi-object fruit images, utilizing k-Means clustering and k-Nearest Neighbor (k-NN) classification methods. This research reported high accuracy levels, achieving 92.5% for single object images and 90% for multi-object scenarios[1].

Another relevant research developed a mobile application for detecting cat breeds through deep learning. This study employed the Mobilenet\_v1 FPN model and recorded an accuracy of approximately 81.74% for single object images and 60% for multi-object detection. The emphasis in this study was on user-friendly application development, optimizing processing times for practical use[2].

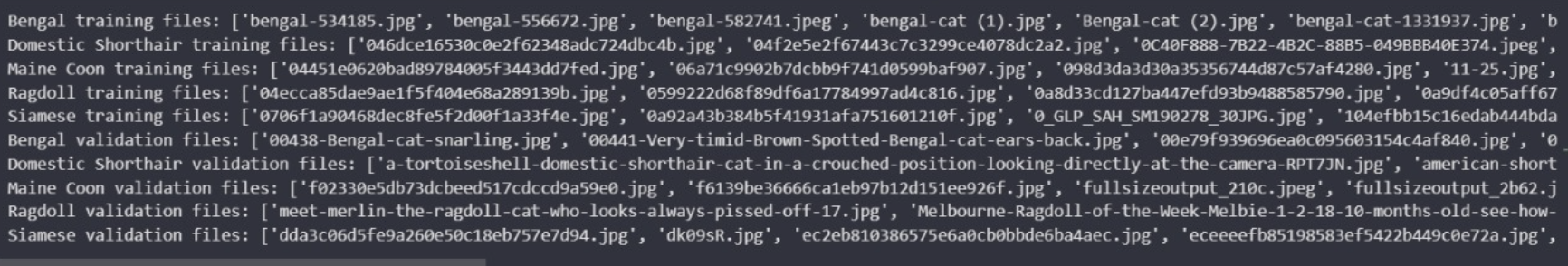
Additionally, the study titled "BVCNN: A Multi-Object Image Recognition Method Based On Convolutional Neural Networks" proposed a novel approach by integrating BING (a segmentation method noted for its speed and efficiency) with vectorization of convolutional networks to enhance processing speed. This method significantly reduced the image recognition time to under one second. However, it faced challenges in multi-object classification due to errors in recognizing objects from target candidate windows, highlighting the complexity and ongoing challenges in this field[3].

## EDA and data preprocessing

In our project on cat breed identification, effective data preparation is crucial for the performance of our convolutional neural network (CNN). We begin by loading the image dataset from a directory, storing each image's file path in a variable. This initial step organizes our data systematically, setting the stage for further processing. Furthermore, in order to have a clear understanding of how many datasets exactly we have, we decided to create a bar chart to depict the data.



After that, we can see through the given output, what breed refers to what image, in order to ensure, that the code accesses the files appropriately.



We began by loading our dataset from a specified directory, storing the path in a variable to ensure easy access and manipulation. The images in our dataset varied in size and pixel dimensions, which required normalization to ensure consistent input for our neural network. Normalizing the images involves scaling the pixel values to a range of 0 to 1, which helps in speeding up the training process and improves the performance of the model by providing a standardized input format.

To further enhance our model's ability to generalize and avoid overfitting, we employed data augmentation techniques during training. Data augmentation artificially creates new training data from existing data through various transformations that simulate different perspectives and scenarios, such as rotations, shifts, and flips. This process increases the diversity of the training examples without the need for additional images, effectively expanding our dataset and introducing more variability.

Firstly, we implemented **Rescaling,** meaning that very pixel in an image was rescaled to a range between 0 and 1. This normalization makes the model training less sensitive to the scale of features.

Later on, we applied **Augmentation Parameters:**

* Rotation: Images were randomly rotated by up to 40 degrees. This helps the model handle different orientations of cats.
* Width and Height Shift: Images were randomly shifted horizontally and vertically by up to 20%. This teaches the model to recognize cats even when they are not perfectly centered.
* Shear Transformations: Applied shear transformations with an intensity of up to 20%. This simulates a tilting effect, helping the model to learn from images where the perspective is slightly distorted.
* Zoom: Random zooming on images by up to 20% to mimic the effect of the cat being closer or further away.
* Horizontal Flipping: Each image was randomly flipped horizontally, a necessary augmentation since cats can face either direction.
* Fill Mode: The 'nearest' fill mode was used to handle newly created pixels after transformations like rotation and width/height shifts.

For validation data, we applied only rescaling to maintain the integrity of the test set, ensuring that the model's performance evaluation is realistic and based on unaltered data. The augmented training data and normalized validation data were then used to feed into our model through generators that handle data loading and augmentation dynamically during model training. This approach not only saves memory but also makes our model robust against variations in new, unseen images.

The code is here below:

from tensorflow.keras.preprocessing.image import ImageDataGenerator

train\_datagen = ImageDataGenerator(

rescale=1./255.,

rotation\_range=40,

width\_shift\_range=0.2,

height\_shift\_range=0.2,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=True,

fill\_mode='nearest')

test\_datagen = ImageDataGenerator(rescale=1./255.)

train\_generator = train\_datagen.flow\_from\_directory(

training\_dir,

target\_size=(200, 200),

batch\_size=25,

class\_mode='categorical')

validation\_generator = test\_datagen.flow\_from\_directory(

validation\_dir,

target\_size=(200, 200),

batch\_size=20,

class\_mode='categorical')

## Modelling

In the development of our project, we utilized a convolutional neural network (CNN) constructed with TensorFlow, a choice driven by the framework's robust functionalities for building and training deep learning models. The CNN architecture designed for this task included several key components aimed at effectively processing and classifying the image data into one of five cat breeds. The model started with an input layer designed to handle images of 200x200 pixels with three channels (RGB). Following this, several convolutional layers were employed, each featuring filters of increasing depth (32, 32, 64, 64, 128, and 256). These layers are crucial as they extract various features from the images at different levels of granularity. ReLU activation functions were used to introduce non-linearity, enabling the model to learn more complex patterns. Max pooling layers followed each convolutional layer, with a pool size of 2x2. These layers reduce the spatial volume of the input image after convolution, which not only helps in decreasing the computational load and the number of parameters but also makes the features detected by the convolutional layers robust to position and orientation changes. After extracting and processing the features, the data was flattened from a matrix to a vector to prepare it for classification. This was followed by two dense layers with 1024 and 512 neurons respectively, both utilizing ReLU activation. These layers serve to interpret the features extracted by the convolutional and pooling layers by generating predictions on the breed classification. The final layer in the model architecture was a dense layer with five neurons, corresponding to the five cat breeds, using a softmax activation function. This layer outputs the probabilities for each class, making it possible to classify each image by the breed with the highest probability. The model was evaluated based on its accuracy, achieving a rate of 65%. This measure reflects the percentage of breed classifications that were correct, providing a straightforward metric of the model’s performance.

You can see the code snippet below:

import tensorflow as tf

model = tf.keras.models.Sequential([

tf.keras.layers.Conv2D(32, (3,3), activation='relu', input\_shape=(200, 200, 3)),

tf.keras.layers.MaxPooling2D(2, 2),

tf.keras.layers.Conv2D(32, (3,3), activation='relu'),

tf.keras.layers.MaxPooling2D(2,2),

tf.keras.layers.Conv2D(64, (3,3), activation='relu'),

tf.keras.layers.MaxPooling2D(2,2),

tf.keras.layers.Conv2D(64, (3,3), activation='relu'),

tf.keras.layers.MaxPooling2D(2,2),

tf.keras.layers.Conv2D(128, (3,3), activation='relu'),

tf.keras.layers.MaxPooling2D(2,2),

tf.keras.layers.Conv2D(256, (3,3), activation='relu'),

tf.keras.layers.MaxPooling2D(2,2),

tf.keras.layers.Flatten(),

tf.keras.layers.Dense(1024, activation='relu'),

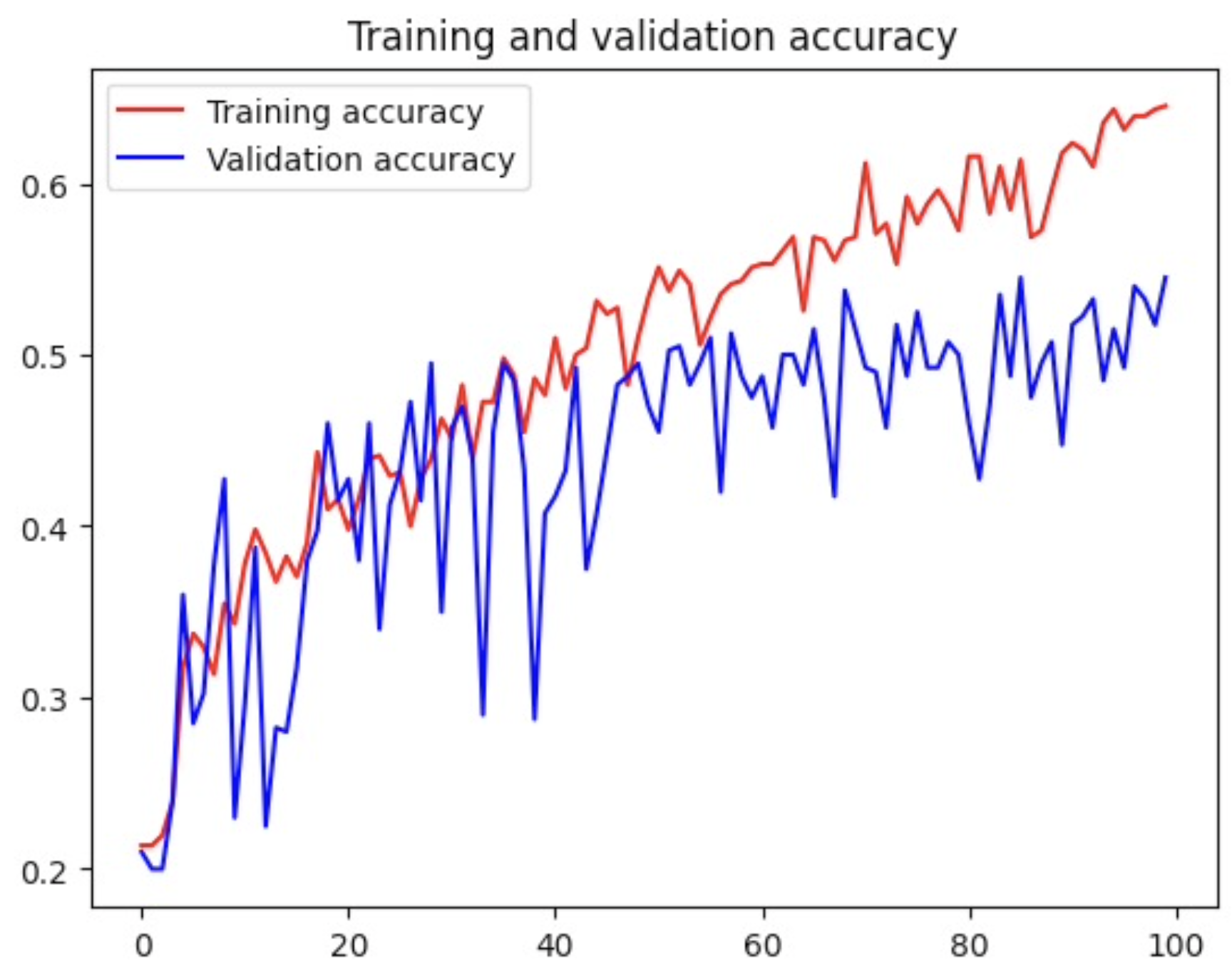
tf.keras.layers.Dense(512, activation='relu'),

# Output layer with 5 neurons for 5 classes and softmax activation

tf.keras.layers.Dense(5, activation='softmax')

])

You may see the visual depiction below:

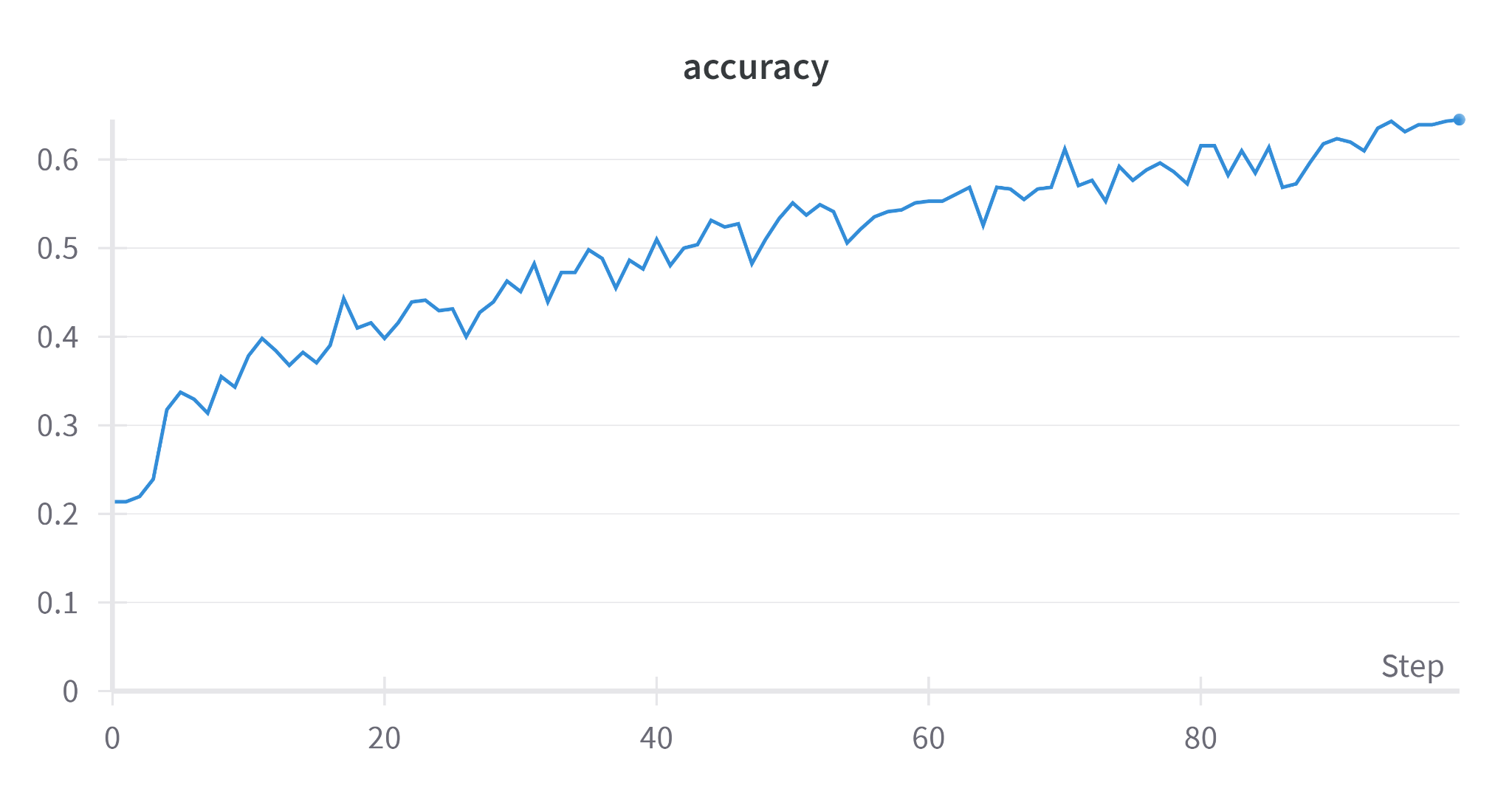


## Experiments

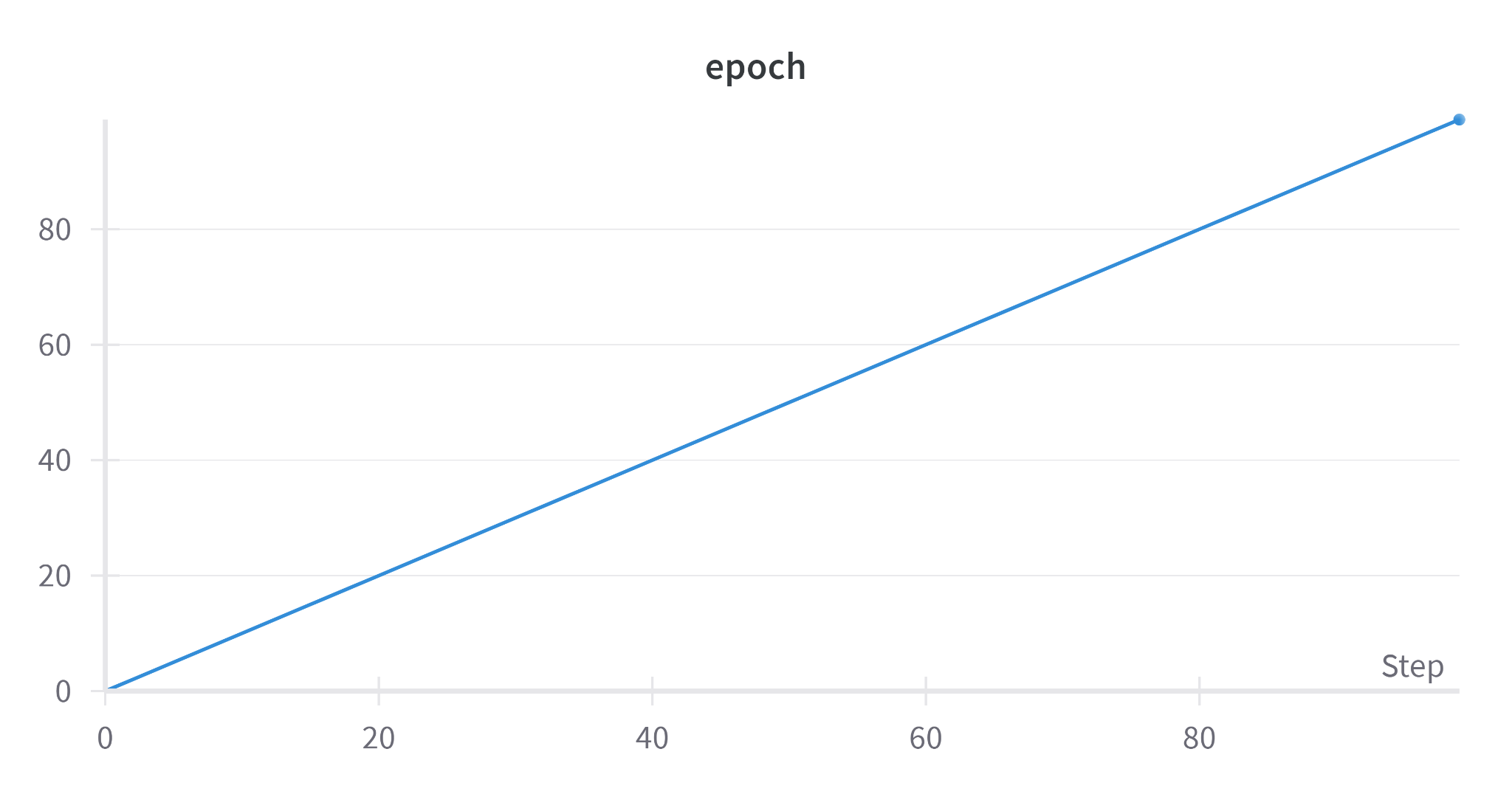
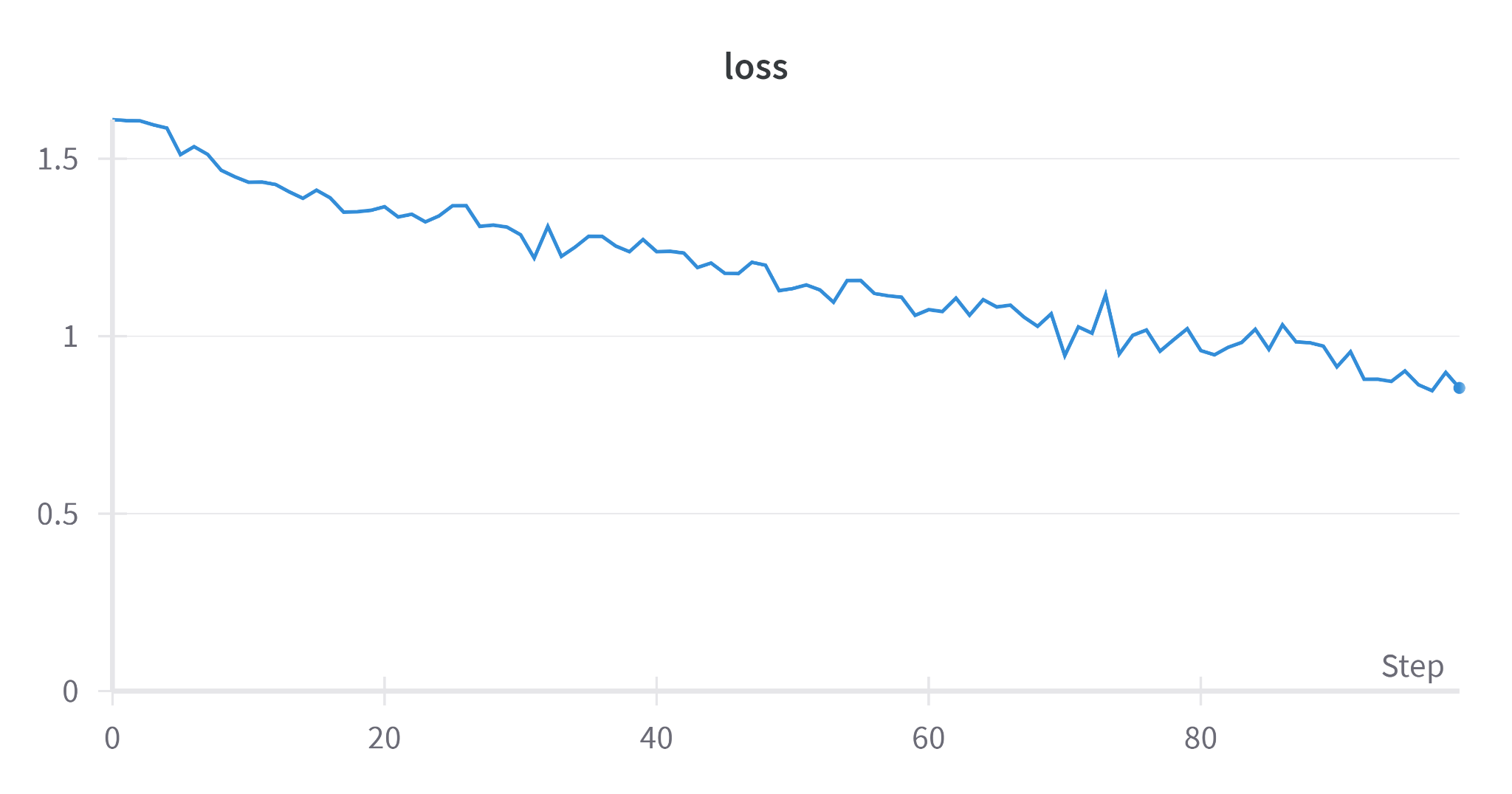
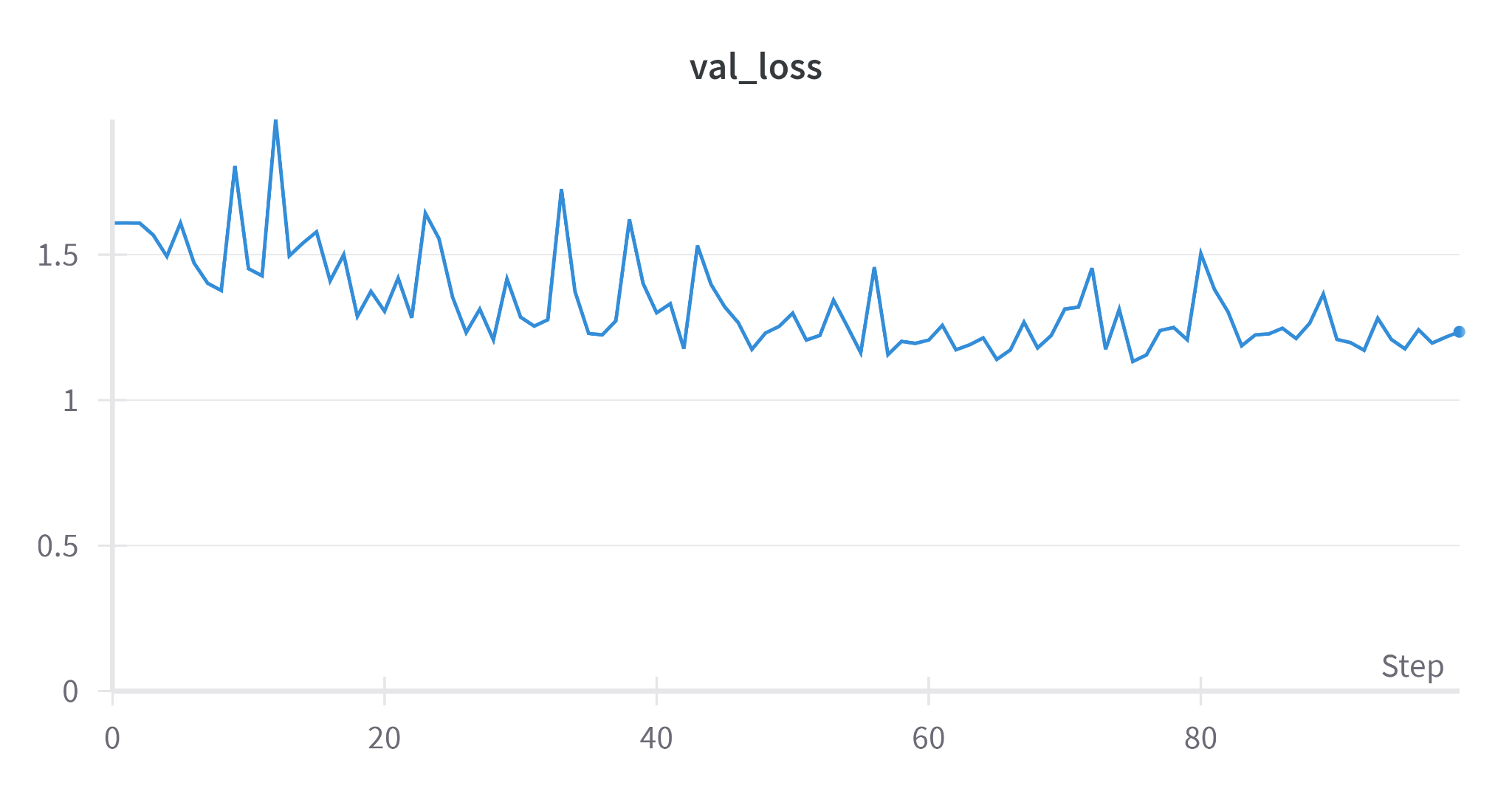
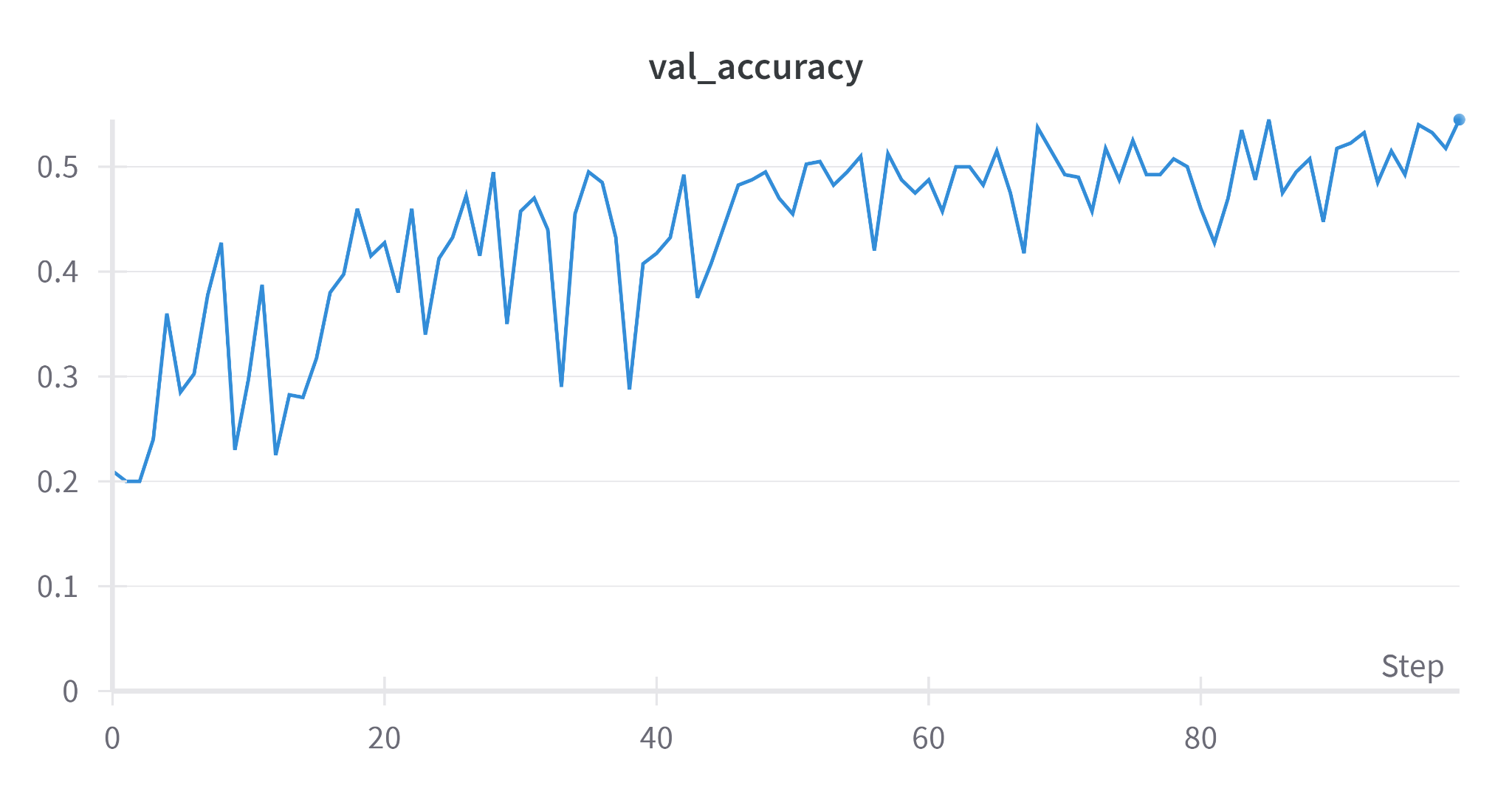
The experiment conducted involved training a deep learning model using TensorFlow for image classification. We integrated Weights & Biases (wandb) as the MLOps tool to monitor and visualize the training process.

Weights & Biases (wandb) have many advantages. Firstly, Wandb seamlessly integrates with popular deep learning frameworks like TensorFlow, PyTorch, and Keras, making it easy to log and visualize metrics during model training. The wandb dashboard provides a comprehensive view of training metrics, including loss, accuracy, and more as inserted below. It allows for easy comparison between different experiments and hyperparameters. Wandb automatically tracks experiments, making it easy to reproduce results and iterate on model improvements. Each experiment is logged with its configuration, code version, and metrics, facilitating collaboration and reproducibility. Wandb offers collaboration features that allow team members to share experiments, insights, and comments, fostering collaboration and knowledge sharing within a team.

On the other hand (wandb) has disadvantages. For users new to wandb or MLOps tools in general, there may be a learning curve in understanding how to effectively use all of its features and integrate them into existing workflows. Since wandb is a cloud-based service, we are dependent on its availability and performance. Connectivity issues or downtime could potentially disrupt the training and monitoring process.



As it is seen from the graph, accuracy is 65%.

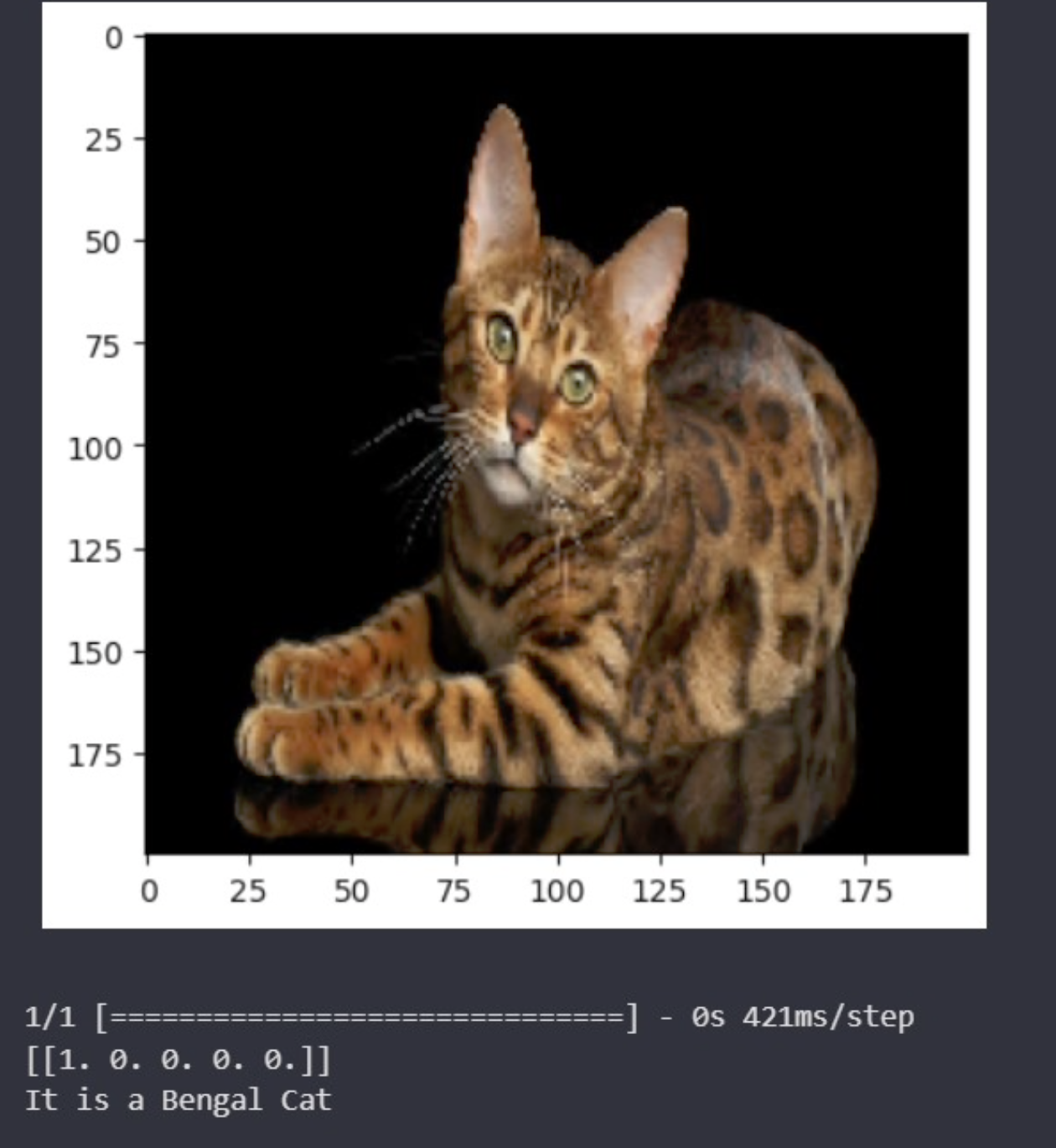


Overall, Weights & Biases (wandb) is a powerful tool for managing and monitoring machine learning experiments, providing valuable insights into model performance and facilitating collaboration among team members. However, users should be aware of the associated costs and invest time in learning how to leverage its features effectively.

## Discussion of results

The performance of our convolutional neural network (CNN) in classifying cat breeds was evaluated using a predefined test set as well as additional, personally selected unseen images. This approach provided a comprehensive understanding of how well the model generalizes beyond the data it was trained on. Achieving a test accuracy of 65% is a promising result, indicating that the model has a reasonable capability to recognize and differentiate between the five cat breeds. However, the accuracy also suggests that there is significant room for improvement. The fact that the model did not achieve higher accuracy could be due to the need for more training data. The performance on unseen images is particularly crucial as it mimics real-world scenarios where the model would encounter images that vary in many aspects from the training dataset.

You may see a sample below:



## 

## 

## 

## 

## 

## 

## 

## 

## References

[1] M. Fachrurrozi, A. Fiqih, B. R. Saputra & R. Algani, “Content Based Image Retrieval for Multi-Objects Fruits Recognition using k-Means and k-Nearest Neighbor,” 1–6, 2017.

[2] X. Zhang, L. Yang & R. Sinnott, “A Mobile Application for Cat Detection and Breed Recognition Based on Deep Learning,” in AI4Mobile 2019 - 2019 IEEE 1st International Workshop on Artificial Intelligence for Mobile, 7–12, 2019.

[3] H. Shi & S. Wang, "BVCNN : a multi-object image recognition method based on the convolutional neural networks", https://doi.org/10.1109/ICVRV.2015.28, 2015.