Faiza Abdullah

Lab 06

ITAI 1378 Comp Vision-Artificial Intelligence

Professor: Anna Devarakonda.

Chihuahua or Muffin: A Reflective Journey in Image Classification

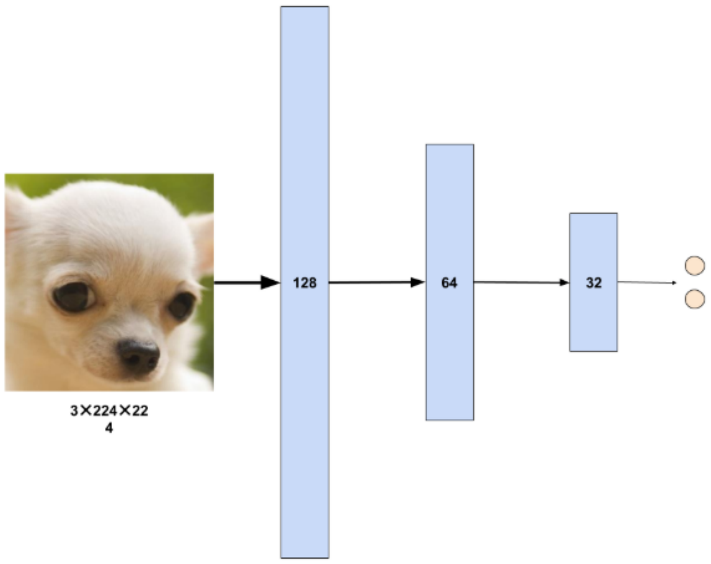
Summary of the Workshop's Main Objectives and Techniques Used

The "Chihuahua or Muffin" workshop, executed in Google Colab, aimed to construct a neural network classifier to differentiate between images of Chihuahuas and muffins—a whimsical yet insightful challenge in image classification.

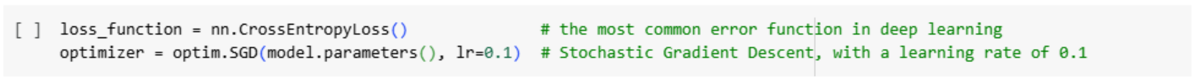
The primary goal was to leverage PyTorch to build, train, and evaluate a simple neural network, using a dataset split into training (120 images) and validation (30 images) sets. Key techniques included defining a custom neural network (‘MySkynet’), preprocessing images with ‘torchvision.transforms’, loading data via ‘ImageFolder’ and ‘DataLoader’, and training the model with cross-entropy loss and stochastic gradient descent (SGD). The workshop progressed through cloning a GitHub repository (https://github.com/patitimoner/workshop-chihuahua-vs-muffin.git), setting up the environment, and visualizing results, all within Colab’s interactive Jupyter interface.

Key Concepts Learned

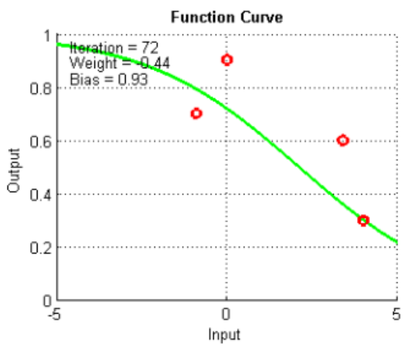
The workshop introduced me to image classification, where a model assigns labels based on visual features. I explored neural networks, specifically fully connected layers in ‘MySkynet’, which flattens RGB image tensors (3×5×5) into vectors and processes them through hidden layers (128, 64, 32 neurons) to a 2-class output.

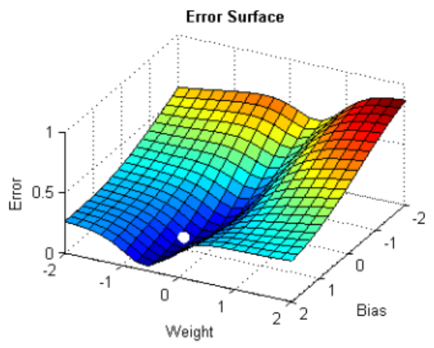


Data preprocessing with transforms like ‘Resize’ and ‘Normalize’ ensured compatibility with the model, while data loading via ‘ImageFolder’ organized the dataset into labeled folders (‘chihuahua’, ‘muffin’). The training loop reinforced optimization concepts, using SGD to minimize loss over epochs, monitored by accuracy and error metrics.



```
[ ]  loss_function = nn.CrossEntropyLoss()             # the most common error function in deep learning
     optimizer = optim.SGD(model.parameters(), lr=0.1)  # Stochastic Gradient Descent, with a learning rate of 0.1
```





Challenges Encountered and How I Overcame Them

A notable challenge arose with ‘image_datasets’ and ‘dataloaders’, where placeholders like "?" in the code confused me:

```
image_datasets = {
   "train": datasets.ImageFolder('data/train', train_transforms),
   "validation": datasets.ImageFolder('data/validation', validation_transforms)
}
dataloaders = {
   "train": torch.utils.data.DataLoader(image_datasets['train'], batch_size=64, shuffle=True,
num_workers=4),
   "validation": torch.utils.data.DataLoader(image_datasets['validation'], batch_size=64,
shuffle=False, num_workers=4)}   # Replaced ?
```

Initially, I didn’t understand why validation data wasn’t shuffled or how to set ‘batch_size’. Debugging in Colab, I printed ‘image_datasets["train"]’ and found it contained 120 datapoints, revealing the dataset structure. Referencing PyTorch docs, I deduced that ‘shuffle=False’ for validation preserved order for consistent evaluation, and I set ‘batch_size=64’ to match training. Switching Colab’s runtime to GPU via “Runtime > Change runtime type” also resolved slow training, enhancing performance.

Insights Gained About Machine Learning and Image Classification

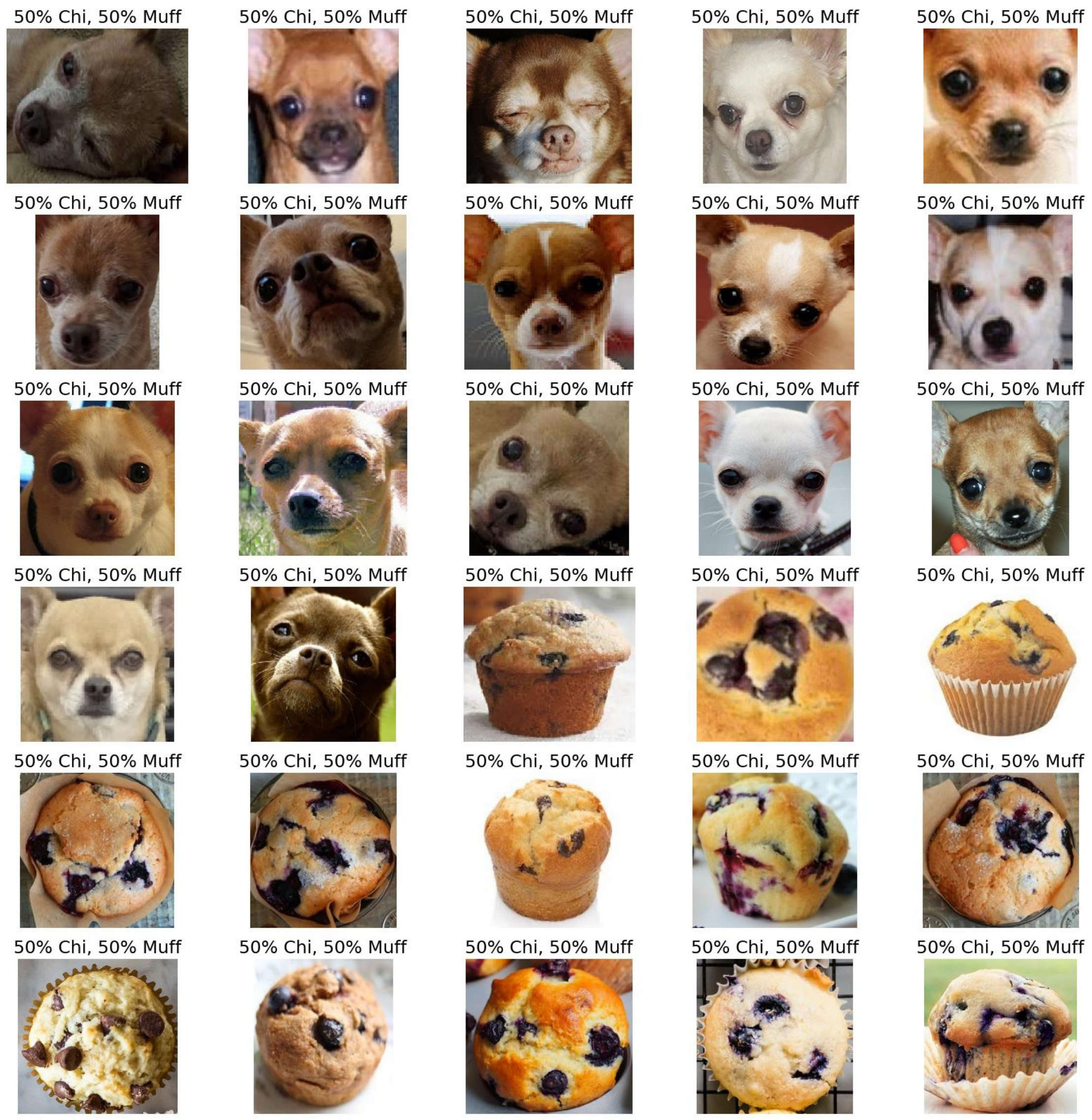
This experience highlighted neural networks’ ability to discern subtle visual cues, yet their dependence on structured data and tuning. The stagnant accuracy (~54%) over three epochs suggested underfitting, possibly due to the small 5×5 input size losing detail or the lack of convolutional layers, unlike advanced CNNs. It emphasized the train-test split’s role in detecting overfitting, as validation metrics gauged generalization beyond rote learning. The workshop bridged theory and practice, showing how loss functions guide learning.

Potential Real-World Applications

In medicine, neural networks classify X-rays for disease detection. Agriculture could use them to identify plant diseases from photos, while retail might tag products visually. The Chihuahua-muffin dilemma mirrors practical tasks like spotting defects in manufacturing or distinguishing species in ecological monitoring, showcasing image classification's versatility.

Personal Reflections on the Learning Experience

Working through this workshop was both rewarding and challenging. Running code interactively in Colab, like cloning the repository and visualizing data, felt intuitive and engaging, especially with the quirky dataset keeping things light. However, the initial ambiguity in code placeholders frustrated me, pushing me to dig into documentation and experiment—a process that built confidence. Seeing the model's modest performance sparked curiosity about enhancing it with CNNs or larger inputs. This hands-on dive into ML ignited my enthusiasm to explore beyond this tutorial, perhaps tackling a personal classification project next.



Citations

https://www.deeplearningbook.org/

https://github.com/patitimoner/workshop-chihuahua-vs-muffin

https://teachingcommons.stanford.edu/teaching-guides/artificial-intelligence-teaching-guide/warming-ai

https://pytorch.org/tutorials/beginner/introyt/trainingyt.html

https://discuss.pytorch.org/t/how-to-split-dataset-into-test-and-validation-sets/33987

https://colab.research.google.com/github/patitimoner/workshop-chihuahua-vs-muffin/blob/master/workshop_1.ipynb#scrollTo=emLe_bPHA3VG https://github.com/AbdullahFaiza/jupyter-exploration/blob/main/Chihuahua%20or%20Muffin.ipynb

https://pytorch.org/docs/stable/index.html

https://datascience.stackexchange.com/questions/111128/understanding-sgd-for-binary-cross-entropy-loss

https://www.geeksforgeeks.org/stochastic-gradient-descent-classifier/

https://paperswithcode.com/task/image-classification

https://levity.ai/blog/image-classification-in-ai-how-it-works

https://www.frontiersin.org/journals/plant-science/articles/10.3389/fpls.2023.1158933/full

https://nhsjs.com/2024/early-detection-of-crop-diseases-using-cnn-classification/