Final Report

CSE 144 Winter 2024

Alison Sun, Harsh Jha
{allasun, hjha}@ucsc.edu

March 20, 2024

1 Division of Labor

Alison: data preprocessing, fully connected layers, evaluated model on testing set, early stopping, CSV file submission, report
Harsh: Resnet transfer model, data augmentation, train and validation run, L2 regularization, dropout layers and batch normalization, adjusted learning rate and weight decay

2 Model Details

This model utilizes the Resnet-50 architecture, with three batches of convolutional layers and batch normalization with ReLU. We utilized the resize, random horizontal and vertical flip, and color jitter methods to augment our image data. To train the model, we froze all its layers save for the convolutional batches and the last fully connected layer, and then gradually decreased the output parameters of the convolutional layers, adding dropout and ReLU after each. Additionally, L2 regularization loss was added as a method to shrink weights and combat overfitting. Finally, the model stops early if validation accuracy does not improve after 5 consecutive epochs.

3 Environments

Our model utilizes a number of Python libraries:

- `numpy` was used to set `best_val_loss` to `np.Inf` while creating and training the model with PyTorch.
- `pandas` was used to generate the CSV file for submission.
- `torch.nn` provides our neural network architecture and methods, including convolutions (`nn.Conv2d`), and ReLU and cross-entropy loss functions.
- `torch.optim` sets our optimizer (Adam, with 0.001 learning rate).

- `models` from `torchvision` allows us to access the Resnet-50 architecture and set its weights.
- `transforms` from `torchvision` is used for data augmentation.
- `datasets` from `torchvision` allows us to access the ImageFolder class.
- `DataLoader` from `torch.utils.data` is used to iterate over training, testing, and validation datasets.
- `tqdm` adds progress bars in the output when we run our model.
- Since we ran our model in Google Colab, `drive` from `google.colab` helped us access our training and testing datasets, which were stored in Google Drive.
- `random_split()` from `torch.utils.data.dataset` is used to split the training data into training and validation datasets (80/20).

4 Usage

Code was run in Google Colab, using a V100 GPU. Training and testing data were connected from Google Drive via the `drive.mount()` method from the Google Colab library.

5 Model Link

The link to our model is: