10805 Mini-Project Proposal

Weiran Yao (wyao1) [1] **Bingqing Chen (bingqinc)** [1] **Arnav Choudhry (achoudhr)** [1]

1. MP-A: musical mode classification on the million song dataset

1.1. Dataset

The Million Song Dataset (MSD) is about 280 GB in size, and was created for the purposes of research on algorithms at-scale and provide a standard dataset for various Music Information Retrieval (MIR) tasks (Bertin-Mahieux et al., 2011). It contains 1,000,000 songs from 44,745 unique artists. For each song, MSD contains audio features such as duration, energy, loudness and meta data such as artist name, title, year of release. Each song has a binary variable for musical mode denoting it to be either major or minor. We are going to use this variable as the target variable.

1.2. Problem statement

The specific modeling question that we aim to solve is to train a classifier to determine the musical mode for a song, given the set of audio features and metadata for that song.

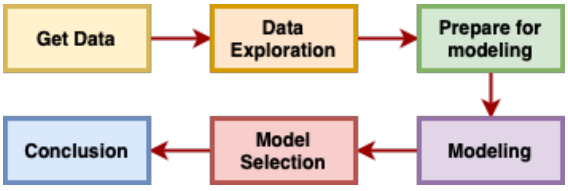

Figure 1. Proposed pipeline for Mini Project A

The proposed steps in our machine learning pipeline are detailed in Figure 1. In the get data step, we first get the raw data as a sequence of SQLite databases and text files, and then we process the data and augment it by linking records from each file to a central SQLite database that would contain all the information. We then perform some data exploration where we focus on finding distributions, correlations and missing values. In the prepare for modeling stage, we first create a tabular dataframe which contains all the features of interest and the labels from the exploration stage along with any transformations, imputations, interactions and (dis-)aggregations of those features. We then split the data into train, test and validation sets.

In the modeling stage we first establish metrics of interest such as accuracy, computational and storage costs. As the next step we plan to train several classifiers and record their performance suing the discussed metrics. We plan to use the logistic regression classifier as our primary classifier, and search over different hyperparameter settings and feature sets as different models. In the model selection stage, we then compare the performance of different models and choose the best one based on the value of the validation metric. We also compare the performance of the selected classifier(s) with naive baselines in this step. And finally we collate the results and create some visualizations to show our findings in the conclusion stage.

1.3. Compute

We estimate how much memory we will use and the computing time using 1 core CPU on Databricks with a subset of the Million Song Dataset. This subset of dataset has 6,724 data points with 12 features.

Table 1. Computing resource estimation on 1 core CPU

	Subset dataset	**Full dataset**
Dataset size	1.3MB	280GB
Memory usage	645KB	≈240GB
Sample size	6,724 samples	≈ 1M samples
Feature size	12	≈3,0000
Model	LinearReg+SGD	LogisticReg+SGD
Batch size	6,724	6,400
Iterations	500	500
CPU hours	5 seconds	≈516 hours

- **Memory usage:** as shown in Table 1, suppose each entry is stored as a double, the full dataset is expected to use 240GB memory.
- **CPU hours:** because logistic regression using gradient descent has the same time complexity as linear regression, we can use it to estimate the computing time for the full dataset. Suppose we still run SGD, which is of time complexity $O(np)$, with 6,400 samples in

[1]Carnegie Mellon University, Pittsburgh, PA 15213.

10805 Class Project, Fall, 2020.

one batch for the full dataset, the same experiments are expected to take 516 hours on a single CPU. However, if feature filtering/hashing can be done before fitting the full dataset, the total CPU hours (1 core) are expected to be reduced to 100 hours;

- **AWS budget:** We plan to use one m5.xlarge machine as master node, two m5.xlarge machines as worker nodes and 100 c5.xlarge (two cores) machines as task nodes. The computation for one experiment is estimated to take 30 minutes and cost $2.57. Suppose we will run 10 experiments and in total the computation will cost **$25** and take **5 hours**. The storage on S3 will cost **$10** per week. The computation cost is within the $100 credits given in the course.

2. MP-B: image classification on CIFAR-100

2.1. Dataset

The CIFAR-100 dataset (Krizhevsky et al., 2009) consists of 60,000 32×32 color images in 100 classes. The dataset occupies about 200MB of disk storage. There are 600 images for each class, which are split into 500 training images and 100 test images. Each image has the `coarse` (superclass) label and `fine` (class) label. We use the 100-class `fine` labels for this project.

2.2. Problem statement

We execute a 100-class classification task in this project. We will implement three convolutional neural networks (CNNs) to compare their accuracy, model complexity, and computational load. Initially, we considered using basic machine learning models, such as logistic regression and random forest. But, it quickly becomes clear to us that these models will not do well, after prototyping and evaluation. Thus, we opted for CNNs.

Models: For estimating the computational load in the proposal, we implemented All-CNN (Springenberg et al., 2014), a simple, light-weight model ($\approx$ 1.3M Parameters). We will compare it to two other CNNs with closer to the state-of-the-art (SOTA) performance (SOT, 2020)

Metrics: Accuracy is defined as the percentage of samples for which the classifier makes the correct predictions, The model complexity is evaluated as the number of learnable parameters in the model. The computational load will be evaluated both in terms of how much GPU memory and how long a model takes during training.

2.3. Compute

We use All-CNN (Springenberg et al., 2014) to estimate the required computation resources, while keeping in mind it is smaller compared to SOTA models. The model is implemented in `PyTorch` and the results are summarized in Table 2. The accuracy over epochs is shown in Figure 2.

Table 2. Summary of Metrics for All-CNN

	All-CNN
Accuracy	70.68% (Train); 55.63% (Test)
Model Parameters	$\approx$ 1.3M
GPU Memory	6267MiB (batch size = 500)
Computation Time	18.3 min (50 epochs)

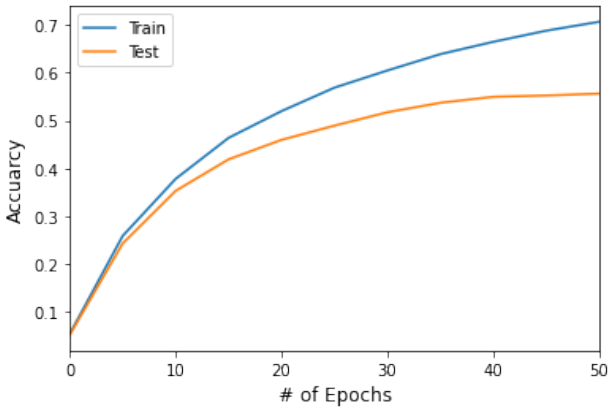

Figure 2. Training and test accuracy over epochs

AWS Budget: Since the dataset is only 200MB, we can train the model on a single GPU machine. While we ran the experiment on a personal GPU machine, it is most similar to AWS `p3.2xlarge`, which costs $ 0.27/hour based on EMR pricing. The storage cost on S3 can be ignored because of the small dataset size. We trained our preliminary model in less than 30 minutes. With consideration of the development time and the two other models being larger, it is still well within our computing budget.

References

Papers with code - cifar-100 benchmark (image classification), 2020. URL https://paperswithcode.com/sota/image-classification-on-cifar-100.

Bertin-Mahieux, T., Ellis, D. P., Whitman, B., and Lamere, P. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.