CS5242 Assignment 3 Report
Terence Kong
E1287893

# Introduction

Weakly Supervised Clustering by exploiting unique class count is a multiple instance learning technique that allows us to make predictions on instances by only using bag level labels. Here we attempt to reproduce the results on the CIFAR10 dataset.

# Implementation

Architecture

The following changes were made to the architecture:

- Batch Normalisation on the encoder output layer to prevent extreme values from being passed to the KDE.
- Removed 1 fully connected layer from the Distribution Regression Module (DRN) as it was not able to learn any features (mostly 0 weights).

Everything else was kept the same. I.e. Using Resnet for the encoder and decoder layers.

Data Augmentation & Regularisation

For the data loader, I went with a different implementation approach as I wanted to explore how varying parameters such as the distribution of instance class types, ucc, variable bag sizes etc. would affect the result. The idea was that making the model more robust to different types of data scenarios would make it a lot more practical for conventional workloads. However, I found that doing so affected the distribution of the estimated probability density functions (PDFs) from the KDE and often resulted in the ucc model not being able to learn any meaningful features from the dataset. Thus, I've reverted to using the same distribution model as the paper (i.e. even distribution of ucc classes and instances)

Apart from the custom bag loader, I've mostly used normalisation for regularisation. As previously mentioned, the distribution of the dataset is important for the KDE. As such, I felt like using excessive data augmentation could potentially make it harder for the model to extract good PDFs from the features.

Parameters

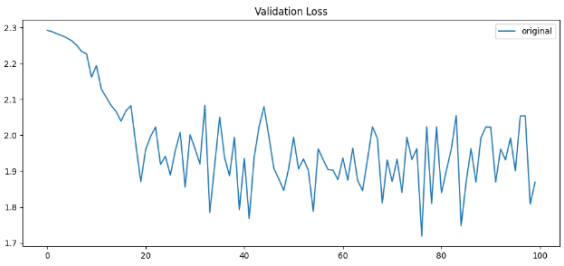
Most parameters are kept the same as in the paper. I slightly tweaked the num_bins and sigma values in order to get a smoother KDE result.

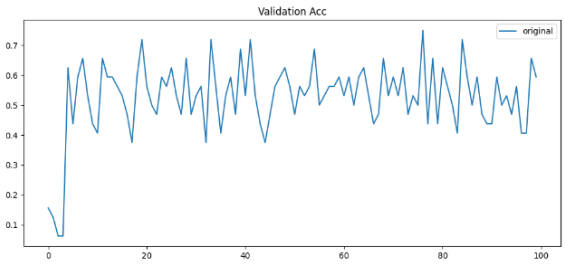
| Parameter | Values | Hyperparameters | Values |
|---|---|---|---|

| | | | |
|---|---|---|---|
| num_features | 16 | Optimizer | Adam |
| num_bins | 20 | Learning rate | 0.001 |
| sigma | 0.05 | Loss Functions | MSE/CE |
| bag_size | 25 | | |

# Results

| | Min JS Divergence | UCC Acc. | Clustering Acc. |
|---|---|---|---|
| UCC | 0.01 | 0.625 | 0.217 |

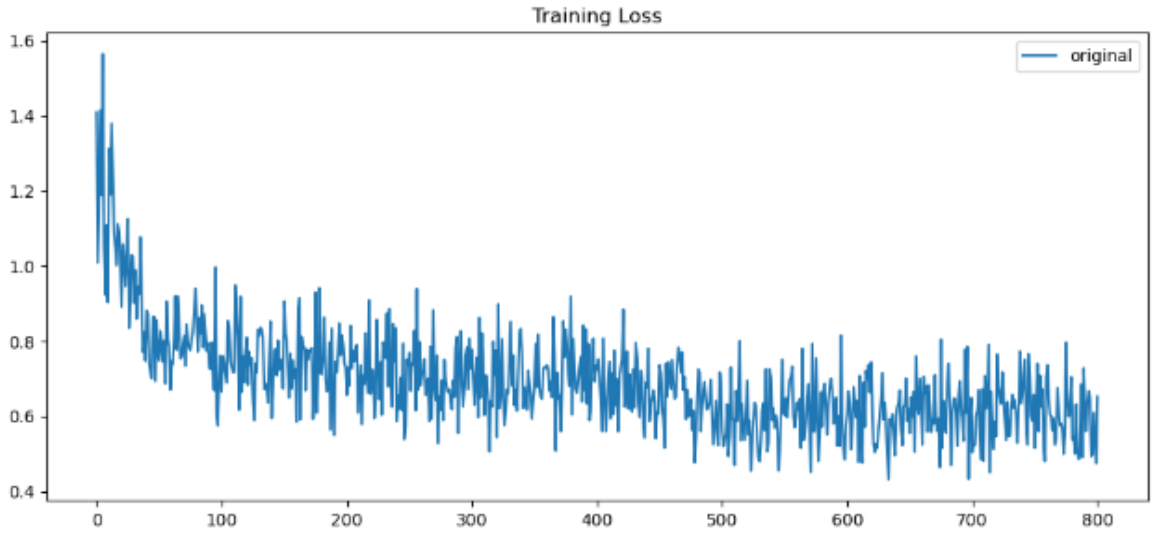



I was not able to replicate the paper's results in my experiments. From the graphs shown, The best I was able to do was 70% on validation accuracy. One of the reasons was due to the fact that the ucc model was particularly difficult to train. Since the model ucc model learns from the KDE and features extracted by the encoder, the two most important areas are the feature extractor architecture as well as the distribution of the dataset.

For the feature extractor, I experimented with various architectures such as MobileNet and WideResnet, but most of them provided pretty similar results but were also much slower to train.

For the distribution of the dataset, how the instances in the bags are organised and also how the bags themselves (and their ucc representations) are organised would affect the distribution of the dataset. I suspect that for this approach to work, a learnable distribution has to be present in the dataset. As such, more stable distributions such as uniform (i.e. even distribution of classes) are much easier to learn than unbalanced distributions.




## Possible Improvements

In the paper, we utilise a Gaussian Kernel in the KDE. However, it might be useful to apply a different kernel for different tasks. The Gaussian Kernel assigns non zero probabilities to negative values, which is fine in this instance since we are predicting ucc values but might not be optimal in other scenarios. Some other kernels that could potentially be useful are:

- Kernels that give less weight to outliers (e.g. Biweight Kernel, Triangular) might be potentially useful for datasets that are noisy.
- Kernels that are computationally less expensive such as (Epanechnikov and Uniform). Considering that the current model takes quite a bit of time to train.