**Yelp Image-to-Rating Interim Report**

G. Lau (1005865921), H. Sahak (1005903710), V. Tran (1004988775)

ECE324, Prof. Guerzhoy

March 13, 2022

**Project Folder Link:**

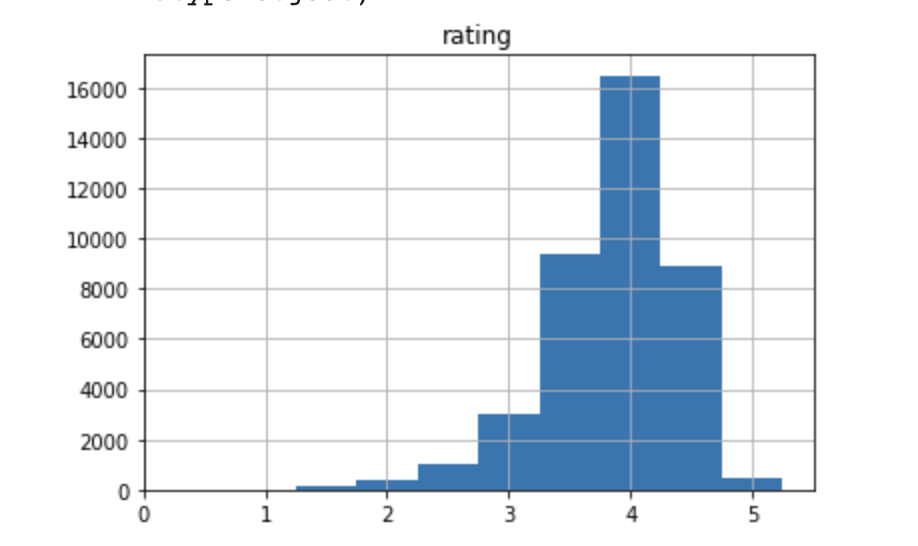
https://drive.google.com/drive/folders/1QnKmXui7nYeHTvnnS6sP\_4\_65uxjcTyj

**Introduction**

This report details the interim progress we have made on our Yelp Image-to-Rating project. Recalling our proposal, the goal of the project is to be able to predict the rating of a restaurant using photos of the food from customer reviews. The group has made progress towards this goal in several areas. First, we created a dataset of 40000 images with associated restaurant ratings and restaurant reviews. Second, we tested the performance of a single layer CNN. Third, we tested more complex CNN architectures and performed hyperparameter tuning to optimize performance. Fourth, we experimented with a pretrained model, preserving its weights and training final linear layers to generate predictions. Lastly we researched ways to achieve semantic space representations of our images (including through the use of text reviews in our training procedure) to achieve more accurate predictions.

**Yelp API Data Collection**

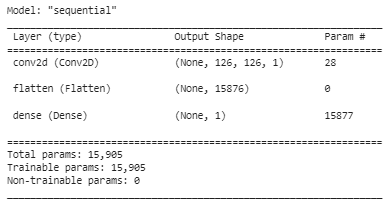
We started by extracting the 40000 images labeled as “food” from the Yelp Images Dataset which contains 200000 images. Yelp also provides the Business\_id of the restaurant where the photo was taken. We wrote a Python script to make calls to the Yelp API to get the restaurant rating for each business\_ID in our dataset. We ran this script over several days due to API call limits and now have 40000 image-rating examples. We wrote and ran a similar script to get up to 3 reviews (and the review-specific rating) for each business\_ID. These reviews may be used in future architectures described in the “Future Architecture Exploration” section of this document.

****

*Figure 1: Histogram of Restaurant Ratings from the 40k image Yelp Dataset*

**Basic CNN Performance**

Before progressing to more complex architectures motivated by literature (e.g., ensemble method), we first trained a very basic CNN to use as a baseline when assessing performance of future networks. This CNN had only one convolution layer, with filter size (3,3) and ReLu activation function. This layer was then flattened and connected to a dense layer with one output that attempts to predict the final rating. See figure below for model summary.



*Figure 2: Simple CNN Model Summary*

Due to limitations on Google Colab, it was infeasible to train on the entire dataset. Our training set was reduced from 40k images to 10k images. Performance obtained on our test set was assessed using the mean squared error. The simple CNN achieved a mean squared error of 0.3249, which performs worse than a network that outputs 4 all the time (mean squared error = 0.3175). This motivates us to build more complex architectures. We start by sticking to a CNN, but making the architecture more complex by adding more convolutional layers and increasing dimensionalities of layer channels, as well as introducing pooling layers. These experiments are discussed below.

**Improving CNN Through Hyperparameter Tuning**

As is normally done with ML models, we perform hyperparameter tuning and select a model that performs best on the test set. Strictly, the validation set should be used to determine which parameters are best, but performance of the selected model is presented by stating results on the test set.

In our case, we performed 6 iterations of hyperparameter tuning. We essentially use the random search technique from ECE421; we change parameters by selecting different hyperparameters and changing their values individually.

**Model 1:**

Instead of having just one convolution layer, we change the architecture to be significantly more complex. We have a 2D convolution layer with 32 channels, 5 by 5 filter and ReLu activation function, followed by a max pooling layer with a 2 by 2 window, another convolution layer with 64 channels, a 3 by 3 filter and ReLu activation function, another max pooling layer (identical to previous max pooling layer), and finally a convolution layer (identical to previous convolution layer). The output is then “fed” to a flattening layer and then a fully connected layer to obtain values for 32 hidden neurons, and finally, a fully connected layer with one output neuron.

This architecture was chosen because it gave good results for the task of classifying CIFAR dataset images. In our case, it gives a mean squared error of 0.5529, worse than just guessing 4 all the time!

**Model 2:**

We suspect that our model may be too complex. So, we make it simpler by removing one of the convolution + pooling layer pairs, and reducing the number of channels from 64 to 8 in the last convolution. Furthermore, we reduce the filter from 5 by 5 to 3 by 3. We do not want to have too big a filter size in the first step of convolution, because we may miss out on localized regions within the pictures that we can learn from. Model2 gives a mean squared error of 0.3013, better than guessing 4 all the time.

**Model 3:**

This is the same as model 2, but we change the stride parameter in the first convolution layer to be (2,2). The stride parameter is the number of pixel shifts over the input matrix when producing the output of a convolution layer. The idea here is to capture more general features, while maintaining locality (3 by 3 filter) and whilst reducing network complexity.

The result of training is a test loss of 0.3134. It turns out the trained network performs worse than model 2 on the test set. However, the deviation is not far off (0.0121/0.3013 = 4%), so we will keep the stride parameter to speed up training in future models, and we can remove it in our final model if it gives better performance for it.

**Model 4:**

We are unsure which activation function is best for our case. The literature seems to suggest ReLu is very popular for CNN training [8], but we will try tanh as it has given us good results in previous CNN tasks (ECE421 assignment). The network trained with tanh activation gives 0.3021 loss. This is better than 0.3134, so we will continue using tanh activation for our convolution layers.

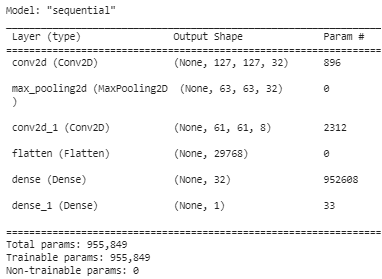
**Model 5:**

The fifth model is obtained by replacing max pooling with mean pooling, or average pooling. This turned out not to affect mean squared error on the test set by any significant value. The mean squared error was 0.3072, which is larger than 0.3021 (model 4), so we will stick with max pooling. This is consistent with many CNN implementations in literature and in practice [9].

**Model 6:**

The last model is obtained by copying architecture for model 4, but changing image size to be 256 by 256. We suspect that bigger image size may lead to learning better representations of the image when reducing dimension through convolution or pooling.

This model gives us our best results- mean squared error of 0.2991 on the test set. Model summary is shown below.



*Figure 3: Best-performing CNN-based model*

We now have a minimum viable product, but need to analyze which test cases our model is performing poorly on, and find ways to combat this. Immediate next steps include obtaining a Google Colab Pro account and producing histograms of results for each rating (to see if we are performing poorly for specific ratings). We also need to explore image augmentation techniques. These are all easily done and made available by common Python libraries such as Tensorflow and Keras [10]. In the following section, we will discuss the possibility of using pre-trained networks as feature extractors, to speed up training and treat our task as a fine-tuning task.

**Pretrained CNN Performance**

This section is work in progress. We are currently in the debugging phase; however, all of the code is already written and available in the notebook titled “Pretrained Networks”. The inspiration for using pre-trained CNNs comes from the paper “*A Deep Learning Ensemble approach to the Yelp Restaurant Classification Challenge*” [proposal]

In this paper; after optimizing a base model, the authors focused on different datasets to pre-train or fine-tune a CNN on. The idea was to extract more expressive features for certain types of images; e.g by using a model used for food when extracting features for a food image rather than relying on ImageNet which is trained on a very general image database. The final result was 3 fine-tuned VGGNet models trained on ImageNet, a places database (MIT’s Places) for scene recognition and a food database (Food-101), to be more representative of Yelp images. Six fusion methods were investigated, the best was found to be late fusion, specifically bagging, which uses a voting procedure such that at least 2 models have to output probability > 0.5 for a class for the ensemble to predict that class. The model achieves an F1 score of 0.82, which is 12% higher than baseline (0.73) and obtained 25th of 355 participants in the competition.

While our task is regression and not classification, it is still worth investigating whether the general ensemble method works for our case. Instead of limiting ourselves to VGGNet trained on different datasets, we will investigate several pre-trained models, including VGG16, ResNet50 and MobileNetV2.

Currently, we are debugging high mse values observed in the VGG16 training but expect completion of these networks by March 15th. Progress is tracked in the “INCOMPLETE- NOT FOR INTERIM” folder.

**Further Architecture Exploration**

The fairly lackluster performance from basic convolutional networks calls for further exploration in the architecture design space. As touched on previously [proposal], we propose a novel processing pipeline that first transforms input images into a sentiment analysis embedding. This embedding can then be fed into a fully connected feedforward network to produce a rating prediction. This architecture takes inspiration from previous work by Tang et. al. in which Yelp and Rotten Tomatoes ratings are predicted from text reviews. A key processing feature in this work is the sentiment vector representation of the text to great effect [Tang].

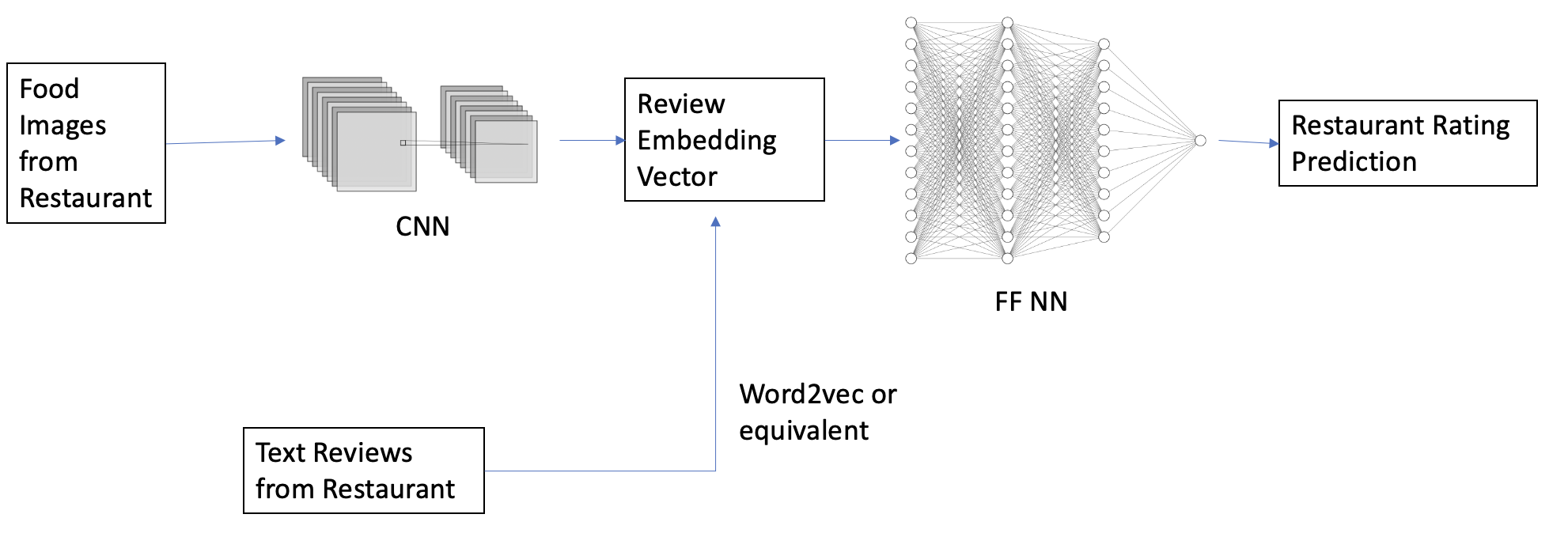
Though CNNs are much more widely known for their use in object classification and segmentation, it has been shown that they can also be used for sentiment analysis of images as well. In particular, work by You et al. illustrates a fine-tuned CNN with transfer learning incorporated that is capable of image sentiment analysis over a training set of weakly labeled Flickr images [Adobe]. Further research pertaining to the sentiment of social media images using various approaches such as colour composition and shape analysis may also prove to be useful [IEEE1][IEEE2]. Deliberate sentiment analysis will play a key role in our pipeline, as it provides a clear direction to achieving accurate rating predictions. Training a network to learn the *emotion* or *sentiment* depicted by an image appears to be more effective than classification on an intuitive level. Indeed, it would make sense that identifying a human sense of deliciousness or disgust from images, for example, could very well be more useful than identifying objects.

To this extent, we take inspiration from DeViSE, a model that places images in semantic space by incorporating the use of both labeled images as well as training with unannotated text. DeViSE was able to glean useful semantic relationship information from its text training, which was then used to improve visual task success rates by 18%. This was done by making a visual network architecture target the specific embedding vectors produced by their text model during training. The primary focus for images was classification. Most notably, however, is that semantic information given by the text-trained component of the model gave DeViSE the ability to make meaningful inferences when classifying images to labels that it was not trained for [google].

These additional works provide extra insight and direction for our next steps in developing a more effective architecture. The Yelp database is rich with text reviews that we can leverage to the same ends as DeViSE, where specific sentiment knowledge learned from reviews can be fed into our CNN as training targets. We have established that CNN sentiment analysis is feasible, and we expect a higher proportion of sentiment/emotion-nuanced context from reviews as opposed to a generic corpus. Our CNN outputs can then be fed into a final set of layers. We intend for this sub-architecture to perform a regression of sentiment features determined by the CNN to a rating from one to five. A disadvantage to leveraging only Yelp review text is a comparatively lower training set size, however. We are presented with three approaches that we can assess moving forwards. A chief subject of interest is whether the use of a large-corpus training method such as the employment of a pre-trained word2vec network, a Yelp-trained network, or large-corpus training with further parameter tweaking using Yelp text is sufficient. Novel approaches to further fine-tune our sentiment targets also exist and may prove to be beneficial to the overall architecture. Specifically, a context-based approach as described by Kim et al. may help deal with concerns of subjectivity, especially in user-written reviews [IEEE3]. Our immediate next step will be to implement common pre-trained models both standalone and as part of our pipeline.

**Diagram of Proposed Model Architecture:**

The image below depicts the general structure of our proposed network architecture. To summarize from the previous section; we wish to learn an embedding from images to embedding vectors that are expected to contain meaning, then pass this through a feed forward neural network to produce a final rating.



*Figure 4: Labeled Diagram of Proposed Model Architecture*

**Bibliography**

[proposal] ECE324 Proposal

[Tang] <https://arxiv.org/pdf/2106.11596.pdf>

[IEEE1] <https://ieeexplore-ieee-org.myaccess.library.utoronto.ca/document/8609672>

[IEEE2] https://ieeexplore-ieee-org.myaccess.library.utoronto.ca/document/7425952

[Adobe] https://arxiv.org/abs/1509.06041

[google] https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/41473.pdf

[IEEE3] https://ieeexplore-ieee-org.myaccess.library.utoronto.ca/document/9366882

[8] <https://ieeexplore.ieee.org/abstract/document/7966185>

[9] <https://onlinelibrary.wiley.com/doi/abs/10.1002/ima.22337>

[10]<https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/>