Image-classification of Asset Management Systems' content based on Machine Learning

Chris Kapp, Ran Song, Karina Rodriguez Echavarria

1 The Problem

In this research project we set out to determine if machine learning could be used to classify a large collection of images into a taxonomy that relates to a specific client or set of clients within an industry. Accomplishing this would help to reduce the reliance on human operators and improve their workflow. While it may not be possible to fully automate this process, utilising machine learning to perform the monotonous tasks would free up humans to deal with more complex or interesting issues. In this particular task, there is an added complexity that some of the clients describe their images in more abstract concepts. For example, a university would find it beneficial to differentiate between images of lecturers and students but may also wish to determine if the students are happy. On the other hand, a shop would look to differentiate between products being modelled rather than the models themselves. While recent advancements in deep learning have provided a solid framework for basic image classification, learning adjectives that describe human emotions or interactions between objects in an image is a much more complex task.

2 Our Approach

To begin, we compiled a shortlist of approaches that we thought may be successful:

1. An unsupervised, clustering based approach similar to the word vector model but for images.

2. A supervised, fine-tuned binary classification approach developing a set of highly accurate yes/no questions to classify the images.

3. A weakly-supervised, fine-tuned, multiclass approach that builds on top of the type of approach used in the ImageNet[2] challenge.

For the clustering approach, we looked to determine if the internal vector representation learned by a model trained on ImageNet model could be used to group the images with similar content via, for example, Euclidean distance. To achieve this, we extracted the 4096-dimensional vector of activations at layer 'fc7' of a VGG19[6] model for a set of images. We hoped to apply dimensionality reduction to this vector to provide a visual representation of the data and an interface for a human operator to select clusters of images to label. Both principal component analysis (PCA) and t-Distributed Stochastic Neighbour Embedding[5] (t-SNE) were tested to perform the clustering. In the case of t-SNE, we would first use PCA to reduce the dimensionality such that it retained approximately 80% of the total variance. While t-SNE did produce reasonable results for a well-defined set of objects like shoes and bags, both t-SNE and PCA results were unusable

for more complex data. We suspect this is likely caused by the relatively low data variance (15-20%) maintained by the algorithms once they had projected down to two or three dimensions. As a result, we decided to avoid pursuing this avenue any further.

Next, we moved onto our second area of research, a tree of binary classifiers. We thought this may produce good results as binary classification accuracy is typically very high, as seen in our results. Unfortunately, this method does not scale as well as we would like across a large taxonomy and would require huge amount of compute power to do so. During this, we tested the performance of several well known deep learning algorithms to determine which performed best on the type of data we were using. The chosen algorithms were Inception-v3[7], ResNet-100[3], and VGG19. While these are no longer state of the art in terms of ImageNet performance, they require significantly less memory to use and would allow us to take advantage of TensorFlow's GPU acceleration. Additionally, TensorFlow Slim has model checkpoints that have been trained on ImageNet that we can fine-tune from with some model surgery, this allows us to benefit from transfer learning. In our testing, we found Inception-v3 to be the best performing algorithm in terms of accuracy, validation loss, and training time thus we chose to continue our experimentation with this algorithm.

Lastly, we investigated the potential of a multiclass approach to address the scalability problem of our binary approach. For this, we fine-tuned the Inception-v3 algorithm to classify two different sets of data. The first set of data is reasonably well defined while the second set of data contains a fair amount of noise. That is to say, some images appear in more than one class. We found that the Inception-v3 algorithm was somewhat resilient to this noise and produced good top-1 accuracy in addition to near perfect top-3 accuracy in both tests. The idea behind this approach would be to fine-tune a model for each group of taxonomies, one per industry. For example, one network that specialises in identifying university-related classes, another in shop-product related classes, etcetera. This would allow us to classify very specific concepts that can have different, but semantically similar, human-readable labels for each individual client.

3 Existing Commercial Alternatives

To determine the validity of our research, we needed to construct a fair method of comparison. We began by researching a few of the existing commercial alternatives and discuss our findings:

1. Google Cloud Vision

   Google Cloud Vision does not currently support training a custom model around a specific data set. This is not as desirable as a specialised model will outperform a generalised model at the task it is designed for and as such will not provide much meaningful insight.

2. Amazon Rekognition

   Amazon Rekognition is a generic image classification system in a similar light to Google Cloud Vision and as such suffers the same problems.

3. Amazon SageMaker

   Amazon SageMaker allows the user to train a new model to fit their data. They also advertise the ability to write your own TensorFlow/MXNet/PyTorch/etc code and use their platform as hosting similar to the other AWS platforms. As they do

not appear to offer a generalised pre-built system, we would just be testing our code on their hardware which would lead to identical results. This bespoke code is likely to give the best accuracy as there is a finer degree of control.

4. IBM Watson

   IBM Watson offers support for custom models, but the free tier is limited to 1000 events per month. This is not really enough to generate an accurate representation of their performance and as such is left omitted from the comparison.

5. Clarifai

   Clarifai let the user process 5000 images for free in a month, and support custom fine-tuned models which makes them an ideal candidate to benchmark against. However, it is noteworthy that this image limit includes both training and testing. Additionally, their API allows for the creation, training, evaluation, and inference of custom models programmatically with just a few function calls. This API is available in several languages including Python, Node, Java, and C#.

6. Vize

   Vize also has a 5000 image per month limit for free users using custom trained models and thus can also be used as a benchmark. However, their REST API is less friendly to use than their competitors library, requiring significantly more code to execute basic commands. Additionally, they do not currently support multi-label classification unlike other services like Clarifai.

4 Comparison Methodology

For our testing, we chose four sets of data:

1. Facial expression binary classification

   This set contains 13,315 images from WaterAid in which we attempt to classify whether or not the persons in the image are smiling or not. We chose this set as it is a more difficult task for computer vision and yet is something that is very intuitive to humans.

2. Well defined objects

   This set contains 8999 images from The Hut Group. Each image contains either a bag or shoes. We chose these because they are very distinct objects that we would expect a neural network to perform very well in classifying, and consequently provide a solid baseline measurement.

3. Multiclass-4

   A set of 22,314 images that belong to one of the four classes mentioned prior, this requires a slightly different approach to classification in the network and typically results in slightly poorer performance than a binary classification approach. However, this modification enables a single model to scale across a taxonomy of potentially hundreds or thousands of labels.

4. Multiclass-6

   Our final testing set, containing 36,100 images. Each image belongs to one of six labels; the four mentioned prior, boy, or girl. We noticed that there was significant pollution between the labels where an image would, for example, appear in both the smiling and boy categories. This is designed to simulate a more real world approach where the data may be noisy and some of the concepts may be difficult to differentiate between.

Due to restrictions in the free trials of each of the commercial alternatives, we randomly sampled 1000 images for training from the training set, and 100 images for testing from the testing set. This allowed us to process each of the four sets for a total of 4400 images to keep within our 5000 image restriction. Additionally, we retrained our models on this smaller 1000 image subset to provide a comparable measure. While this does not necessarily provide an exact measure of how well the networks would perform if fully trained on all the data, it does create a level playing field. These results can be compared to determine the efficacy of each of the approaches on each type of data set. Therefore, as we would expect to see a roughly equal improvement in each model as the number of training images increases, it is logical that the relative performances of each approach should be similar on the full set.

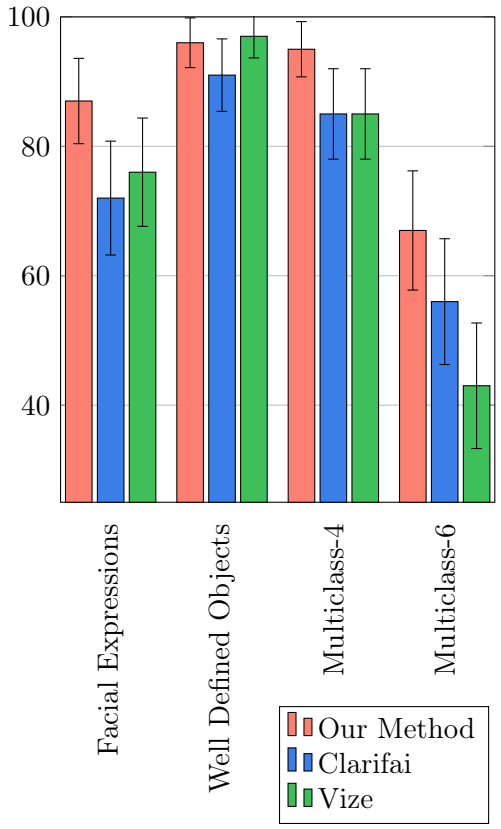

Figure 1: 95% Confidence Intervals for Top-1 predictions

5 Comparison Results

The results of the comparison follow, the top-1 and top-3 accuracies of each method were tested where applicable. The top-1 accuracy represents the percentage of images for which the prediction with the highest score matches the label for the image. The top-3 accuracy represents this same percentage when one of the three most likely predictions from the network matches the label.

As can be seen in Table 1, our method significantly outperformed the existing competitors in all but one test, in which it performed 1% worse than Vize.

To further verify our findings, we calculate the 95% confidence interval for these measurements. By treating each classification in the testing set as a 1 or a 0, we can convert our results into a small population. If the network outputs the correct prediction it is marked as a 1, otherwise it is marked as a 0. Using this data we can calculate a range for which, roughly speaking, we can be 95% certain that the true mean of the data points exists within this range. This helps to distill some of the intrinsic variation contained within neural networks as a result of the optimisation process. Additionally, it helps us to reason about statistically significant differences between the methods based on how much overlap exists between two ranges.

From the information displayed in Figure 1, we can draw some conclusions as to the efficacy of our method. In the facial expression and multiclass-4 data sets, we find a significant jump in mean, and very little overlap in the confidence intervals. This leads us to believe that our method is definitely better in these tasks. For the well defined objects set, we can conclude that Clarifai

Dataset	Accuracy Measurement	Our method	Clarifai	Vize
Facial Expressions	Top-1	**87%**	72%	76%
Well Defined Objects	Top-1	96%	91%	**97%**
Multiclass-4	Top-1	**95%**	85%	85%
	Top-3	**100%**	**100%**	91%
Multiclass-6	Top-1	**67%**	56%	43%
	Top-3	**98%**	90%	85%

Table 1: Table of results

was definitely the worst performer and that our method and Vize perform very similarly. Therefore, our 1% loss is likely a result of the particular minima that was found in this training cycle. Finally, we find a large gap in mean between each of the methods in the multiclass-6 set. While there is some overlap, it suggests that our network is likely also much more suited for this task. Additionally, we note a very low score for Vize in this test despite the network typically having very high confidence in its prediction (>90%).

In this case, we found the use of transfer learning to be paramount to the method. Training our model from scratch saw our top-1 accuracy fall by 37%, with top-3 accuracy falling by 8%. In addition to this, the model took 3.6 times longer to converge at 36 epochs from 10 using ImageNet weights as our initialisation. Without more knowledge as to the processes that Vize and Clarifai are using behind the scenes, it is difficult to conclude the exact causes for our performance improvement. For example, Clarifai provide no further information than 'We developed a proprietary, state-of-the-art neural network architecture, and trained the network over billions of training samples to offer world-leading models for our customers'. However, we hypothesise that one factor may be that they are intialising their custom models with weights from their bespoke prebuilt models rather than ImageNet, which could impact the performance slightly. Another factor could be due to the method of image augmentation used to artificially increase the size of the training set while making the network invariant to changes in the image that do not affect the label, for example a change in brightness or a horizontal flip. Additionally, differing algorithms and hyperparameters such as the learning schedule could also be responsible.

Overall, we conclude that our method matches or significantly outperforms the existing alternatives across the board, most notably in the tasks that are typically more difficult for machines to classify. As this was a focus for this project, we believe these are particularly important categories. For example, our method was very resilient to the noise present in the multiclass-6 set, and handled the multiple faces in many images in the facial expression set very well.

6 Implementation of our method

Our method was implemented using Python and the TensorFlow[1] library, written primarily in Python and C++. We utilised several NVidia GTX 780Ti's to hasten the training process as training on a GPU is much faster than a CPU. Additionally, we make use of the ImageNet pretrained Inception-v3 model from the TensorFlow

Slim research repository for transfer learning. Using this model, we remove the final 1×1 convolutional layer and replace it with our own so that we can fine-tune to our specific classes. We freeze the weights in the stem, and train only from the first Inception module. We split the data into training, validation, and testing sets with a 70/10/20 split respectively. We train on the training data, periodically testing on the validation set to check we are not overfitting the training data. At the end of each pass through the data, or epoch, we compare the performance on the validation set. If it has worsened then we decay the learning rate by a factor of 10, if we dropped the learning rate after the previous epoch and we have still not improved then we deem the model fully trained. We save this model state so that we can benchmark the performance using the testing data set.

In terms of future expansion and improvement of the model, the weakly supervised approach begins with one trained model on the data available at the current time. As this model sees use and classifies new images, we can retrain the model using the, now larger, classified data set. Alternatively, with some code modifications it would be possible to fine-tune the model on the fly but adds some complexity to ensuring the model is not getting worse by doing so. We believe this could be best achieved by designing an interface that allows the user to upload several images in a batch. This interface can then call, for example, a REST API that calculates the model predictions for each image and returns the top n predictions, where n is a proportion of the total number of classes in that taxonomy. This data could then be further verified by a human operator if necessary, for example the system may warn the user if the network has a particularly low confidence in it's prediction and ask the user to select the relevant labels. An algorithm to solve this may involve comparing the predictions starting with the highest probability and checking if the percentage difference between it and the next highest probability is under a certain threshold.

For the infrastructure required to use such a model, a machine with several GPUs would be ideal. For example, each GPU can be designated to run a model related to a specific industry. This would allow very few physical machines to host the system, but training the models could be sped up significantly by parallelising across several GPUs. For example instead of training on batches of images sequentially, one batch can be sent to each GPU and the gradient updates can be collated, averaged, and pushed back to each GPUs copy of the model. This can achieve near linear scaling with the number of GPUs used. These models can be trained and used on the CPU, but this does take significantly longer and it would be very slow to train multiple models at once on one machine. Additionally, in terms of performance per unit cost the multi-GPU solution should be cheaper and more cost effective than multi-CPU.

The time required to design and implement a useable interface will vary depending on the required features. In addition to this, the time taken to train the models will largely depend on the available hardware and number of images in the set. Our multiclass-6 test using all 36,000 images (25000 training images) took approximately 3 hours to train to model convergence on a single GTX 780Ti. This process will be much faster in larger batches on a GPU with more memory. With more modern hardware and some optimisations to the input pipeline to make reading data from disk faster, this process may be possible in under 2 hours. These numbers should scale linearly with the number of images used so it could be expected for a model on similar hardware to

train for about 24 hours per 200,000 training images. However, it is possible that this may require a different number of epochs to converge which will affect training time. Inference for a batch of images (the same as the batch size used for training) on a trained model will likely take in the region of 200-400 milliseconds and as such is likely not an issue.

In summary, a web interface could be paired with a REST API to submit images to a server that can gather predictions from the relevant model. It could then automatically classify images that it has high confidence for, and prompt the user to verify the lower confidence images. These tags can then be saved with the image metadata in a database for later retrieval. Additionally, it may be helpful to store the prediction probability with the image such that this can be used as an additional factor for determining the most relevant images to the end user's query. However the exact methodology for this would depend on the existing technology stack.

7 Recommendations

Our recommendation to take forward the method tested for tagging all images from clients, we recommend a series of steps:

1. Creating generic/sector specific taxonomies and gather content for these (with permission of clients).

2. Exploring whether generic objects can be classified using existing systems. In principle this is possible based on the results shown in section 5, but this will be less accurate than re-training the network.

3. Training multi-class models for more specific terms in the taxonomy. There is a need for further exploration of maximum number of classes which can be incorporated in one model.

4. Creating interfaces to query the network over the Internet.

5. Developing usable interfaces to tag and browse content.

	Low StdDev	High StdDev
Min StdDev	0.2176	0.3397
Max StdDev	0.2484	0.3521
1 label	60	90
2 labels	39	9
3 labels	1	1
Boy	22	6
Girl	40	12
Smiling	20	21
Not Smiling	14	9
Shoe	3	33
Bag	1	19

Table 2: Label noise with respect to the standard deviation (StdDev) of the trained networks output. This shows the number of images by category and by label within the 100 highest and 100 lowest standard deviations of the network outputs.

With regards to infrastructure, it is possible to use systems such as Amazon Sagemaker to host the sytsem and train custom models. It is also possible, to develop the company's own infrastructure (hardware and software).

8 Further Research

In this section we discuss some of the potential research areas that may provide an improvement to our results.

An Evolutionary Approach to Cleaning Labels

Our findings also pose a question as to whether the clustering method we explored might be combined with our final methodology to further improve our accuracy. We hypothesise the existence of a connection between noisy labels and their neighbours in high dimensional space. That is to say, as noisy labels are inherently misclassified, we may be able to detect their presence by looking for outliers at the layer just before the final fully connected layer. If there is an image that is very close in Euclidean distance to several other images and it is labelled as, for example, a dog, but it's neighbours are all cats then we may be able to alter the label. We could achieve this using a k-nearest neighbour method if the network also has a low confidence in it's prediction. To gather more insight into the viability of this method, we define a small experiment. As the internal vector representation can be mapped to the networks output vector we can use the output vector to perform a simplified experiment. By calculating the standard deviation of the output vector of each image in the testing set, we can order the images by the networks confidence. The more uncertain the network becomes of it's prediction, the closer the output of the softmax for each element in the vector tends to the mean, or $\frac{1}{n}$ where n is the number of classes. This, in turn, results in a low standard deviation. Therefore, we sort the testing images by standard deviation and determine how many of the images are poorly labelled, and the distribution of the labels within the highest and lowest 100 standard deviations. As can be seen in Table 2, we find a potential correlation between the poorly labelled images in the testing set and the standard deviation of the output vector of the model.

This cross validation technique may help to remove some of the label noise, thus increasing the accuracy of the model, and could even be applied to the whole data set in a k-fold cross validation technique. If we train k models to calculate this information for the whole data set, using the rest of the set as training data, we could remove some proportion of the low standard deviation examples, α. This value α may differ between data sets, but could theoretically be learned using, for example, a genetic algorithm to remove varying α sections of the data and retrain the models. Removing data from the low end of this distribution removes a larger percentage of dirty labels than exists for the whole data set. Ideally, fine-tuning this α could clean the optimal amount of data automatically, such that we find a local maxima for the percentage of clean labels within the largest subset of the data, ultimately improving the accuracy of the models trained from this subset.

Binarised Normed Gradient Object Detection and Semantic Descriptions

An alternative approach that may see improved results, in a commercial sense, would be to turn the multi-class problem into a multi-label problem via the use of BING objectness detection. The useful bounding boxes extracted from the algorithm could generate a larger data set that would allow us to classify more than one object per image, and assign several labels as a result. An unlabelled image could be passed through this algorithm to sample n images for the model to evaluate. These evaluations of the objects in the image could be used either as direct labels, or as information for a secondary network to construct a description of the image. A network like this is more complex, but there is existing research surrounding it's application to well known datasets[4]. These types of semantic descriptions of an image are more difficult to train

and produce but could be very helpful for the end users.

We attempted to develop some preliminary results for this method using the Python version of the OpenCV implementation, but it did not provide a score to filter the bounding boxes. Unfortunately we did not have enough time to reimplement the BING algorithm or create an interface for the other existing C++ implementations.

References

[1] Abadi, M. et al. [2015], 'TensorFlow: Large-scale machine learning on heterogeneous systems'. Software available from tensorflow.org.
URL: *https://www.tensorflow.org/*

[2] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. and Fei-Fei, L. [2009], Imagenet: A large-scale hierarchical image database, *in* 'Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on', Ieee, pp. 248–255.

[3] He, K., Zhang, X., Ren, S. and Sun, J. [2016], Deep residual learning for image recognition, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 770–778.

[4] Karpathy, A. and Fei-Fei, L. [2015], Deep visual-semantic alignments for generating image descriptions, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 3128–3137.

[5] Maaten, L. v. d. and Hinton, G. [2008], 'Visualizing data using t-sne', *Journal of machine learning research* **9**(Nov), 2579–2605.

[6] Simonyan, K. and Zisserman, A. [2014], 'Very deep convolutional networks for large-scale image recognition', *arXiv preprint arXiv:1409.1556* .

[7] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z. [2016], Rethinking the inception architecture for computer vision, *in* 'The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)'.