PREDICTING TYPICALITY RATINGS USING CAPSULE NETWORKS

A PREPRINT

Shreyas Chandrakaladharan
Center for Data Science
New York University
sc6957@nyu.edu

May 13, 2019

ABSTRACT

Convolutional Neural Networks have long been the standard for Computer Vision. They excel at Object Recognition and Categorization, often on par with human performance. [12] From the perspective of Cognitive Psychology, it is interesting to compare the representations learned by these CNNs with the representations learned by humans. One of the tasks that human representation facilitates is predicting category typicality, an important concept in the psychology of concepts. Lake et al. found that CNNs too predict Category Typicality Ratings quite well. [1] However in 2017, Hinton suggested that CNNs maybe fundamentally flawed by design and proposed a new architecture 'Capsule Networks' [6]. Inspired from that, this project expands on Lake et al.'s work [1] and evaluates the ability of Capsule Networks to predict category typicality. Our results show that Capsule Networks predict category typicality better, though slightly, than CNNs on the MNIST dataset.

***K*eywords** Capsule Networks · Convolutional Neural Networks · Typicality

1 Introduction

A major breakthrough in Artificial Intelligence was the Convolutional Neural Network (CNN). In 2012, CNN reduced the error rate in the ImageNet Challenge, a benchmark for Visual Recognition, by half. It marked the beginning of the wave of Deep Learning. Since then, CNNs have revolutionized the field of Computer Vision and are now the standard for Object Recognition and Categorization. Categorization is an interesting problem in Cognitive Psychology, and CNNs are the leading models to represent categorization by the human brain. A seminal paper 'Deep Neural Networks Predict Category Typicality Ratings' by Lake et al. [1] evaluated how CNNs could predict category typicality ratings and helped push our understanding of the psychology of categorization. This work is an extension inpsired from the same.

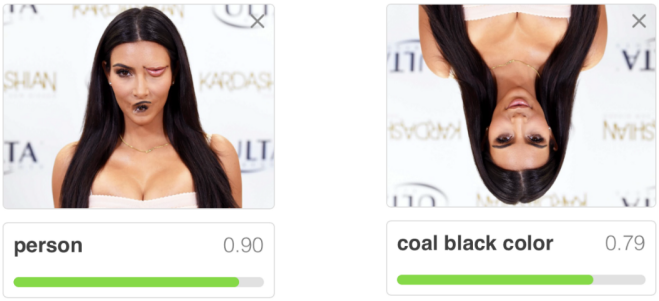
In 2017 Prof. Hinton, one of the fathers of Deep Learning, pointed out a major flaw in CNNs. CNNs, while detecting features in an image, do not account for the spatial locality of the features. For example, operations like max pooling (which acts as an intermediate layer in most CNN architectures) lose spatial locality information. This makes the CNNs susceptible to mistakes such as classifying a morphed facial image as a person. See Figure 1a. Moreover, CNNs also struggle to correctly classify upside down images. See Figure 1b. In contrast, humans do not suffer from such problems. We can easily classify 1a as not a face and 1b as a face. Therefore, CNNs cannot be the pinnacle for representing the psychology of categorization. Interestingly, Prof. Hinton also suggested a radical new approach to Object Recognition called Capsule Networks [6].

Capsule Networks replace the building blocks of CNNs i.e convolutional kernels with Capsules. Capsules overcome the weaknesses of CNNs by encoding information as vectors instead of scalars. Thus Capsule Networks are able to encode more information and establish hierarchical pose relationships which help the network discern the likelihood, position, orientation and size of any object it recognizes. Capsule Networks also replace max pooling operation with

an advanced routing procedure called Routing by agreement [6] which is based on the Expectation Maximization algorithm. This ensures that Capsule Networks do not lose any information due to sub sampling. Of course, the disadvantage of Capsule Networks is the computational cost. Due to vector operations, they are computationally more expensive to train/predict than CNNs.

In this exploratory project, we will use Capsule Networks to predict typicality ratings of digits from the MNIST dataset and compare it with the typicality ratings obtained with CNNs. We will be investigating whether Capsule Networks are closer to human ratings than CNNs. There has been no work done studying the ability of Capsule Networks to predict typicality ratings. So our project is a novel effort at the intersection of Capsule Networks and Cognitive Modeling.

If indeed, according to this study, Capsule Networks are able to predict category typicality of digits better than CNNs then this might suggest Capsule Networks form better representations of digits than CNNs which encourages further research into analyzing if Capsule Networks are a better model for the psychology of Object Recognition and Categorization.



(a) CNN classification mistakes (b) CNN classification mistakes

Figure 1: CNN classification mistakes [9]

1.1 About Capsule Networks

Capsule Networks are convolutional in the sense that they use convolutional layers to extract feature maps from an input image. However, after extracting the feature maps Capsule Networks reshape them into a set of vectors for each location. For example, a convolutional layer with 256 filters extracts 256 scalars as features for each location. This can be reshaped into 8 8-dimensional vectors for each location. Each set of 8 8D vectors is referred to as a Capsule. Furthermore, these vectors are squashed so that their norm is less than 1. This is done to ensure the norm of these vectors is capable of representing probabilities. These capsules generated by reshaping convolutional feature maps compose the first layer of the Capsule Networks known as the Primary Capsule Layer.

In CNNs, after extracting feature maps, a common practice is to use max pooling, a form of subsampling, to induce translational invariance in the network. However, this is the main flaw of CNNs. Instead of translational invariance, we should strive to model equivariance which implies that the network should adapt to different rotations of the input image instead of accounting for all possible rotations of the input image. Capsule Networks achieve equivariance by representing features as vectors and using routing by agreement instead of max pooling [6]. Routing by agreement is explained in more detail in the following paragraphs.

The second and final layer in the Capsule Network, called the Digit Capsule Layer is composed of 10 vectors where each vector represents one digit. The dimension of each vector is a parameter of the network, 16 in our case. Each capsule in the Primary Capsule layer predicts the vector for each of the digits in the next layer via an affine transformation. The final output vector for each digit is a weighted combination of all the capsules in the Primary Capsule Layer. The matrices associated with the transformations are considered as parameters of the Capsule Network. Thus, if there are N capsules in total, there are 10*N matrices. Each matrix is of dimension a*b where a is the dimensionality of each capsule in the Primary Capsule layer and b is the dimensionality of each vector in the Digit Capsule Layer.

We use Routing by agreement to link the Primary Capsule Layer and the Digit Capsule Layer. Routing by agreement is an iterative procedure based on the Expectation Maximization algorithm. At each round, it gets the current predictions from the primary capsules, compares it to the digit capsules and increases the weight of the best predicting primary capsules. The algorithm proceeds for 4-5 rounds and terminates. Now, the length of each output vector in the Digit Capsule layer represents the probability of detecting that digit. Each dimension of the output vector is considered to

represent one aspect of the image like skew, thickness, etc. The sum of all probabilities in the final layer need not sum to one as we are not applying a final softmax operation. The advantage of Routing by agreement is that it induces representing part-whole relationships between the Primary Capsule layer and the Digit Capsule layer. This implies that segmenting an image is more natural and unambiguous.

Finally, a decoder network is attached on top of the Digit Capsule layer. This decoder network tries to reconstruct the input image and adds a reconstruction loss penalizing for failing to reconstruct properly. This acts as a regularizer and prevents overfitting on input images.

2 Method

We asked people to rate a collections of images for category typicality, and we tested three models: Capsule Network, CNN and SVM on their ability to predict these ratings. We used images from the MNIST dataset of handwritten digits. MNIST dataset is ideal for our experiment because:
a) It is a relatively small dataset. Training on it is still computationally feasible with the modest resources available for this project.
b) It is a robust dataset for comparing human and machine learning since writing is a very natural cognitive task. [3]
We will train a vanilla CNN, a Capsule Network and an SVM on MNIST data to predict typicality ratings. Human typicality ratings on MNIST dataset will be collected through online surveys. We will compare and study the correlations of our different models' predictions with the human ratings.

2.1 Computing Resources

MacBook Pro, 2.6 GHz Intel Core i7 with 16 GB 2400 MHz DDR4

2.2 Computational Experiment

First, we trained our three models on the MNIST dataset. The data and the models are described below.

2.2.1 Data

The MNIST dataset has 70,000 <image,label> pairs where each image represents a digit corresponding to its label. For the Capsule network and the CNN, the dataset was split into three sets. 60,000 pairs for training, 5000 pairs for validation and hyper-parameter tuning and the remanaining 10,000 pairs as out of sample test data. We used the default splits provided by TensorFlow [14]. For the SVM, we used a 3-fold cross validation over the entire train and validation data.

2.2.2 Models

- **Capsule Network**

  We implemented the Capsule Network using [11] as reference.

  The first layer, Primary Capsule layer, is composed of 32 maps of 66 capsules each, where each capsule will output an 8D activation vector.

  The second layer, Digit Capsule layer, contains 10 capsules (one for each digit) of 16 dimensions each.

  The decoder network on top of the capsule network is a regular 3-layer fully connected neural network. Refer Figure 2 for the architecture. For more information about the layers, loss functions and the working of Capsule Networks, refer Appendix.

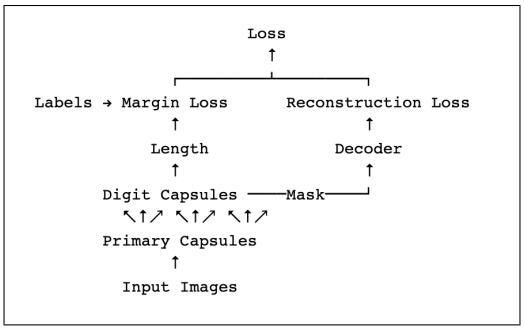


Figure 2: Caps Net Architecture.

- **Convolutional Neural Network**
  Our CNN's architecture is as follows:
  - Convolutional Layer: 32 Filters. 5*5 Kernel
  - ReLU Activation Layer
  - Convolutional Layer: 64 Filters. 3*3 Kernel
  - ReLU Activation Layer
  - Fully Connected Layer: 1024 Neurons
  - Dropout Layer(p=0.25)
  - Fully Connected Output Layer: 10 Neurons
  - Softmax Layer

  We implemented the CNN using [10] as reference. We used the Estimator framework provided by TensorFlow.
- **SVM**
  For our SVM model, we used the SVC implementation from sklearn. The SVC uses a One-Vs-Rest decision function with a RBF kernel. We enabled class membership probability estimates from the SVC.

2.2.3 Training

- **Capsule Network**
  We trained the Capsule Network for 5 epochs. It achieved 99.3% accuracy on the test set. Training took around 8 hrs. Training Capsule Networks takes significantly longer due to number of computations involved. [8] This is also one of the reasons we worked with the MNIST dataset which was possible to train on with our modest computational resources.
- **Convolutional Neural Network**
  We used a Cross Entropy loss and a learning rate of 0.001. We ran the training for 10000 steps with 128 batch size. We achieved 0.0001 loss on training and 99.2% accuracy on the test set. Training took around 10 mins.
- **SVM**
  We used Grid Search with a 3-fold cross validation on a subset of the data (1000 samples) to find the best parameters. Then we fit the best parameter SVC on the entire dataset. The final test accuracy was 97.2% The best parameters were C (Error Penalty)=5.11 and Gamma (Kernel Coefficient) = 0.001. Training took around 20 mins.

2.2.4 Analysis

After training all three models, we wanted to estimate the typicality ratings given by models on the MNIST test set. Following Lake et al.'s work [1], we assume that typicality rating is proportional strength of the model's classification response. Thus, the classification response probability of each model is a good estimate of the typicality rating by the model. For a given image of a digit, our model calculates the probability of the image belonging to each of the 10 possible classes. The classification response probability of the model is defined the probability of the image belonging to the actual class of the image. For example, if our model predicts that the image of a 6 has the label 6 with a probability of 0.98, then the model's classification response probability for that image is 0.98 Accordingly, we extracted the final probability estimates of the MMNIST test set from all the models

In case of CNNs the probability estimates exhibit contrast effects. That is, the model penalizes the example for scoring highly in classes other than its predicted class. This is due to a softmax operation at the end of the final activation layer. To study this effect, along with the probability estimates we also extracted the raw scores from CNNs as the activation of the final layer before the softmax operation.

For SVMs, probabilistic calculations are not natural. We had to manually enable probability estimates in our model. Even these estimates, obtained by Wu et al.'s (2004) extension of Platt Scaling [16] have certain theoretical issues. Moreover, these are also susceptible to contrast effects similar to CNNs. A more natural estimate for SVMs is based on the distance from the margin defined by its support vectors. These are called confidence scores. So, while extracting the probability estimates we also extracted the confidence scores of the SVM using the decision_function API of our sklearn SVM model. These are referred to as the raw scores of SVMs henceforth.

Capsule Networks however do not suffer from such contrast effects. In Capsule Networks the probabilities across all classes in the final layer need not sum to one. Each class's probability is independently calculated. Hence, there was not a need to analyze for contrast effects with Capsule Networks. Thus, we only extracted the probability estimates from the Capsule Network.

Based on the probability estimates from the three models, we defined a good estimate signifying high typicality as being above 0.99 and a bad estimate signifying low typicality as being below 0.8. These thresholds were determined by analyzing the box plots of probability estimates followed by a round of manual verification.

2.3 Stimuli

For our comparison study, we randomly chose 15 good images which had high estimates from all three models (estimate > 0.99) and 15 bad images which had low estimates from all three models. (estimate < 0.8). Typicality ratings were collected from human subjects for these 30 images from the MNIST test dataset. Since these images are from the test set, they were not used for training any of the models. So, the models are not biased towards these images.

2.4 Behavioural Experiment

The 30 selected images were randomly divided into two sets of 15 images each. The participants were presented with both sets. Each set was in the format of a quiz designed with Google Forms. There were 15 questions in each set. Each question had one image and its label. The participants were asked to answer each question based on how well the image fit their idea or image of the category (label). The answer was a rating in the scale of 1 to 10, with 1 being the least typical and 10 being the most. We received 40 responses for each set on average. Human typicality rating for each image was defined as the average of all the responses for that image.

3 Results

We calculated the Pearson's and Spearman's correlation between the human typicality ratings and the models' classification response probabilities. We also calculated the correlations between the human ratings and the models' raw score responses for CNNs and SVMs.

For probabilistic results, Pearson's correlation seems to be more applicable than Spearman's correlation. Spearman's method gives 0.6 correlation to Figure 3c which is clearly not indicative of the actual correlation. Pearson's correlation seems to be more relevant for the probabilistic plots. Comparing the probabilistic results, Caps Net clearly outperforms the other models. Refer Table 1. Our Caps Net achieves a remarkable performance compared to other models as it gets a Pearson's correlation of 0.6 compared to CNN's 0.14 and SVM's 0.43.

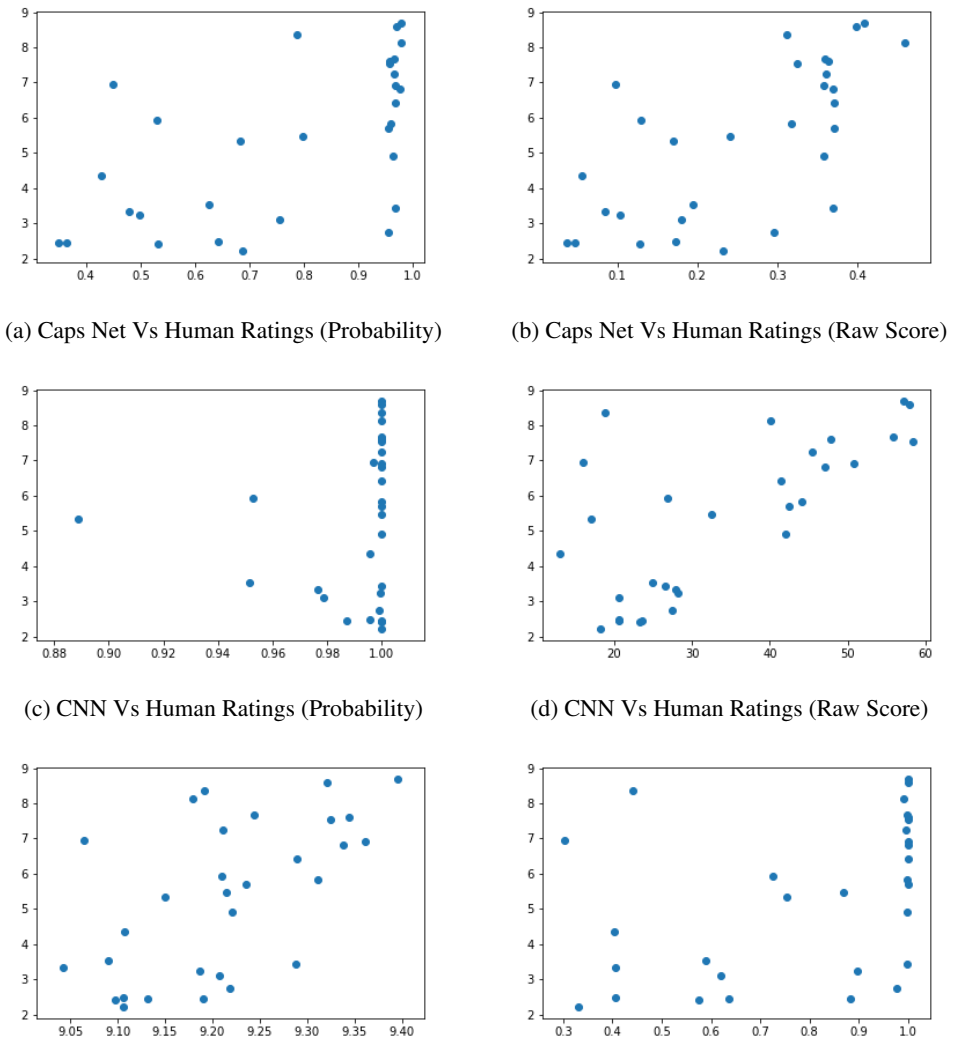
When considering contrast effects and comparing raw scores instead of probabilities, CNNs provided better correlations (Pearson's = 0.68, Spearman's = 0.61) compared to SVMs. These correlations were better than the correlations obtained by Caps Net with probabilistic results. This made us investigate further into the activations of Caps Net. Though the probabilities predicted by Caps Net need not sum to 1 across all classes, we suspected there were some contrast effects. The activations of Caps Net are 16-dimensional vectors for each class with the length of the vector representing the probability of class membership. Each of these 16 dimensions is theorized to be a latent variable of the image representing attributes like shape, size, skew, thickness, etc. Taking that into account, not all dimensions should be useful for classification. Some dimensions might even represent contrast effects. Subsequently, we tested whether a transformation of any subset of these 16 dimensions could provide better correlations. Surprisingly, the norm of a subset of these dimensions provided much better correlations. The norm of the 6 dimensional vector representing the 5^{th} dimension to the 11^{th} dimension of the 16 dimensional activation vector gave 0.67 Pearson's correlation and 0.66 Spearman's Correlation.

Table 1: Probabilistic Results

Name	Pearson's ,p-value	Spearman's ,p-value
Caps Net	0.6003 , 0.0004	0.6458 , 0.0001
CNN	0.1412 , 0.4565	0.6133 , 0.0003
SVM	0.4376 , 0.0155	0.5586 , 0.0013

Table 2: Raw Scores Results

Name	Pearson's ,p-value	Spearman's ,p-value
Caps Net	0.6744 , 4.3e-05	0.6685 , 5.3e-05
CNN	0.6862 , 2.8e-05	0.6106 , 0.0003
SVM	0.5939 , 0.0005	0.5768 , 0.0008



(a) Caps Net Vs Human Ratings (Probability)

(b) Caps Net Vs Human Ratings (Raw Score)

(c) CNN Vs Human Ratings (Probability)

(d) CNN Vs Human Ratings (Raw Score)

(e) SVM Vs Human Ratings (Probability)

(f) SVM Vs Human Ratings (Raw Score)

Figure 3: Graphical Representation of Results

4 Conclusion

Considering classification response probabilities as a good estimate for the typicality ratings, Caps Net are able to explain typicality better than CNNs and SVMs on this dataset as per this study. However, raw score estimates drastically improve the CNNs and SVMs ability to explain typicality . We believe this is due to elimination of contrast effects where a prediction is penalized for being similar to another class. In CNNs, the softmax operation explicitly exhibits contrast effect. In SVMs, the method of calculating probabilites, Platt Scaling, also induces contrast effects. Though Caps Net do not explicitly exhibit such contrast effects on the final layer, we have shown that transformation of the final activation layer of the Caps Net can improve their explanations of typicality as well. This might suggest some dimensions in the final activation vector of the Caps Net might represent contrast effects. While this might be coincidental and inconclusive, this definitely warrants further research into the dimensions of the Caps Net activations and prompts testing Caps Net on bigger datasets such as the ImageNet or Fashion MNIST. Though Caps Nets have a significant computational barrier now, as history has shown, it is only a matter of time before that barrier is broken down. Projects like this could help with breaking the barrier by getting more insights from Cognitive Modelling to simplify operations in Capsules.

5 Future Scope

Thanks to Rueben Feinman for these insights.

Our calculation for raw scores for Caps Nets involves a data leakage. Since we are using our test human data and searching for the best subset of dimensions of the Caps Net activation response, we will not generalize to other human data well. To overcome this, we need to collect more human data and split it into Validation/Test sets. After which we need to find the best subset based on the validation set and calculate the correlation with the test set. Moreover, instead of just finding subsets we could build a Linear Regression Model for predicting typicalities using the final layer activations of Caps Net as input. This can account for more diverse combinations of the dimensions of the final activation response from the Caps Net.

Also, as stated in [1], "There are reasons to suspect that convnets may not see the same typicality structure in images that people do. First, the model parameters are trained strictly to optimize its ability to predict category labels, as opposed to predicting missing features or building a generative model of the data. It may be hard to learn prototypes with this objective". When we train a classifier for digits, we are training the classifier to predict the conditional probability P(Y|X) where X is the given image and Y is the label we want. However, when we ask people to rate the typicality of an image, we are asking them the conditional probability P(X|Y). To best represent this conditional probability, instead of classification models, we need to build generative models. Thus, a potential future scope could be to build Variational Autoencoders, Gaussian Mixture Models, etc for studying their performance on typicality ratings. For example with GMMs, we could compare kernel densities of the different Gaussian clouds to compare typicalities. Additionally we could train these models with new loss functions like Kullback-Leibler divergence loss which are better suited for comparing probability distributions. We could also explore building a new generative model architecture using insights from the theory of capsules.

References

[1] Lake, B. M., Zaremba, W., Fergus, R. and Gureckis, T. M. Deep Neural Networks Predict Category Typicality Ratings for Images. In *Proceedings of the 37th Annual Conference of the Cognitive Science Society*, 2015.

[2] Lake, B. M., Salakhutdinov, R., Tenenbaum, J. B. Evaluating (and improving) the correspondence between deep neural networks and human representations. *arXiv:1902.03477*, 2019.

[3] Lake, B. M., Salakhutdinov, R., Tenenbaum, J. B. Human-level concept learning through probabilistic program inductions. *Science, 350(6266):1332-1338*, 2015.

[4] Peterson, J., Abbott, J., Griffiths, T. Adapting Deep Network Features to Capture Psychological Representations. *Presented at the 38th Annual Conference of the Cognitive Science Society*, 2016.

[5] Peterson, J., Abbott, J., Griffiths, T. Evaluating (and improving) the correspondence between deep neural networks and human representations. *arXiv:1706.02417v3*, 2018.

[6] Sabour, S., Frosst, N., and Hinton, G. E. Dynamic Routing Between Capsules. *arXiv:1710.09829*, 2017.

[7] Sabour, S., Frosst, N., and Hinton, G. E. Matrix Capsules with EM Routing. In *ICLR*, 2018.

[8] Mukhometzianov, R., and Carrillo, J. CapsNet comparative performance evaluation for image classification. University of Waterloo, ON, Canada.

[9] Bourdakos, N. Understanding Capsule Networks. https://medium.freecodecamp.org/understanding-capsule-networks-ais-alluring-new-architecture-bdb228173ddc.

[10] Damien, A. Convolutional Neural Networks with TensorFlow Estimators. https://github.com/aymericdamien/TensorFlow-Examples/.

[11] Geron, A. Capsule Networks Implementation. https://github.com/ageron/handson-ml/blob/master/extra_capsnets.ipynb.

[12] Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks University of Toronto, Canada. 2012.

[13] LeCun, Y., Bengio, Y., Hinton, G. Deep learning. Nature, 521(7553), 436–444, 2015

[14] TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[15] Wu et al. (2004) Probability Estimates for Multi-class Classification by Pairwise Coupling

[16] Platt. C.J (1999) Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Models Microsoft Research, Redmond

This project report was compiled using LaTeX. The code can be found here.