UNDER REVIEW

RESCIENCE C

Replication / NeurIPS 2019 Reproducibility Challenge

[Re] Improved Calibration and Predictive Uncertainty for Deep Neural Networks

Aditya Singh[1] and Alessandro Bay[1],

[1]Zebra Technologies, USA

Edited by
Koustuv Sinha

Reviewed by
Anonymous Reviewers

Received
15 February 2020

Published
–

DOI
–

Abstract

Miscalibration of a model is defined as the mismatch between predicting probability estimates and the true correctness likelihood. In this work, we aim to replicate the results reported by [1] on their analysis of the effect of Mixup [2] on a network's calibration. Mixup is an effective yet simple approach of data augmentation, which generates a convex combination of a pair of training images and their corresponding labels as the input and target for training a network. We replicate the results reported by the authors for CIFAR-100 [3], Fashion-MNIST [4], STL-10 [5], out-of-distribution and random noise data. Our implementation code can be found at https://github.com/MacroMayhem/OnMixup.

1 Introduction

1.1 Calibration

Modern neural networks are miscalibrated, i.e. their predicted confidence value is not reflective of its confidence in prediction. If a model is overconfident, it becomes prone to making wrong predictions with high confidence, thereby depleting the trust on its predictions. It becomes really important in high risk applications (such as in medical diagnosis, automated navigation, etc.) for the neural network's prediction to be correct as well as trustworthy. If the network generates reliable predictions, it would enable the use of some form of fall back mechanism, such as a human-in-the-loop for life critical scenarios. Finding a solution to this problem of miscalibration hence becomes important due to the widespread applicability of deep neural networks across multitude of domains. Figure 1 displays this phenomenon where the average accuracy and average confidence is computed per interval bin (see section 2 for details). The top row (a–e) corresponds to a training scenario where no Mixup is used. As the training progresses the network tends to become overconfident in its predictions. On the other hand, by training with Mixup, the network is much better calibrated, as can be seen in the bottom row (f–j). Note that the network used to generate the results in fig 1 is different from the original paper, which uses a VGG-16 architecture. However, the underlying message i.e. over confident predictions are avoided when using Mixup is conveyed nonetheless.

1.2 Risk Minimisation and Mixup

Given an input $\mathcal{X}$ and an output $\mathcal{Y}$ we are interested in finding $f \in \mathcal{F}$ which describes the mapping from input to the target output. The loss function ℓ penalises the differences

Correspondence should be addressed to Aditya Singh (aditya.singh@zebra.com)
The authors have declared that no competing interests exist.
Code is available at https://github.com/MacroMayhem/OnMixup. – SWH swh:1:dir:824bff001a04ed1c4bb78dc6fbed52fb4470b7a5.
Open peer review is available at https://openreview.net/forum?id=JhZOkalsiI.

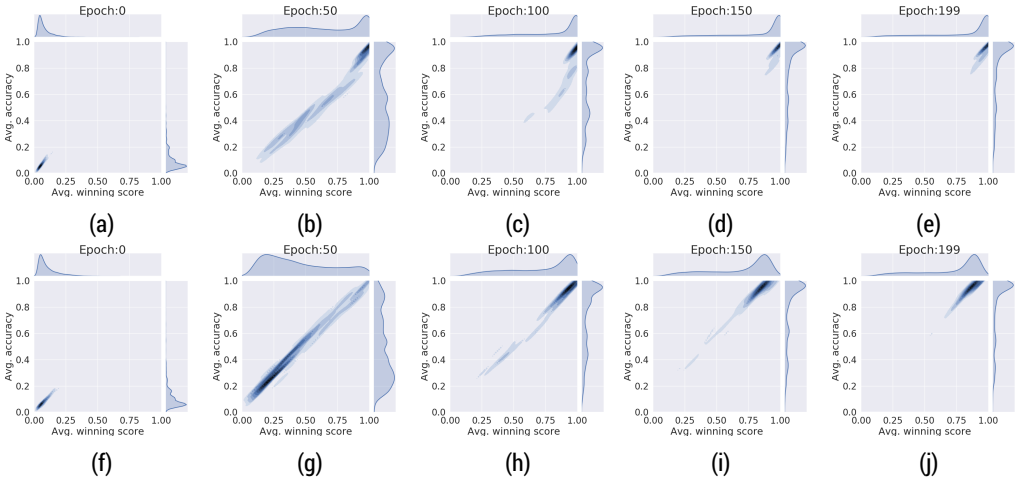


Figure 1. Joint density plot for average accuracy vs average confidence computed for different epochs on CIFAR-100 test dataset using Resnet-34 neural network. Top Row (a–e): Trained without Mixup, the network grows overconfident as the training proceeds, as represented by the sharp peak for the confidence. Bottom Row (f–j): Mixup allows the network to be better calibrated in the corresponding epochs.

between $f(x)$ and y for samples $(x, y) \sim P(\mathcal{X}, \mathcal{Y})$. We aim to minimise the expected risk $\mathcal{R}$:

$$\mathcal{R}(\ell) = \int \ell(f(x), y)\, \mathrm{d}P(x, y). \tag{1}$$

In most scenarios the joint probability distribution $P(x, y)$ is unavailable, and, subsequently, the task is altered to minimise the risk over the training data $\{(x_i, y_i)_{i=1}^n\}$. Therefore, P is replaced by the empirical distribution $\hat{P}$, and this approach is referred to as Empirical Risk Minimisation (ERM) [6]. As a result of this substitution, equation (1) can be modified to find an approximate mapping $\hat{f} \in \mathcal{F}$ for f, as

$$\hat{\mathcal{R}}(\ell) = \int \ell(\hat{f}(x), y)\, \mathrm{d}\hat{P}(x, y), \tag{2}$$

where

$$\hat{P}(x, y) = \frac{1}{n} \sum_{i=1}^{n} \delta(x = x_i, y = y_i), \tag{3}$$

and $\delta(x = x_i,\ y = y_i)$ is a Dirac delta distribution centered at x_i, y_i.
The vicinal probability distribution P_v is the probability of finding a virtual feature-target pair $(\tilde{x},\ \tilde{y})$ in the vicinity of the original pair $(x_i,\ y_i)$. Since the support of $\hat{P}(x, y)$ is a one-point set, it fails to approximate $P(x, y)$ if P is a continuous distribution [7, 8]. Vicinal Risk Minimisation (VRM) [7] assumes that the input distribution is smooth in the vicinity of x_i, and replaces $\hat{P}$ by the vicinal probability distribution P_v, where

$$\mathcal{P}_v = \frac{1}{n} \sum_{i=1}^{n} v(\tilde{x}, \tilde{y} \mid x_i,\ y_i). \tag{4}$$

Mixup [2] is based on the principle of VRM, and generates the vicinal input and the corresponding target from a convex combination of a pair of original inputs and targets. Formally, the vicinal input and output can be represented as

$$\tilde{x}_i = \lambda x_i + (1 - \lambda) x_j \tag{5}$$

$$\tilde{y}_i = \lambda y_i + (1 - \lambda) y_j, \tag{6}$$

where x_i and x_j are randomly selected input samples, y_i and y_j are their corresponding target values, and $\lambda \in [0, 1]$ is drawn from a symmetric *Beta distribution* $\mathcal{B}(\alpha, \alpha)$. The expected risk $\hat{R}_v$ for Mixup can thus be defined as

$$\hat{\mathcal{R}}_v(\ell) = \frac{1}{n} \sum_{i=1}^{n} \ell(\hat{f}(\tilde{x}_i), \tilde{y}_i). \tag{7}$$

1.3 Proposed Method

Mixup serves as simple yet effective data-augmentation procedure to enhance the performance of deep neural networks. The working of Mixup has been proved empirically via extensive experiments to suggest that its contribution in boosting the performance is significant. However, moving away from analysing the boost in performance or explaining why Mixup works, the authors explore the impact of Mixup on the calibration of deep neural networks. The paper performs considerable experiments across a number of datasets to measure the degree of miscalibration with Mixup and make a comparison with clearly defined baselines.
As part of the replication track of NeurIPS Reproducibility challenge 2019, we aim at reproducing the reported results of their hypothesis, i.e. training with Mixup leads to better calibrated neural networks. For the baseline, though not required by the track, we use training without Mixup. The report is structured as follows: Section 2 contains the description of the metrics used by the authors, in Section 3 we provide a detailed information of the implementations which we gathered from the authors' submission as well as email correspondences. We also state clearly where we make certain reasonable assumptions in implementation. Section 4 contains the results on various datasets. In the end, we provide our concluding remarks in Section 5.

2 Calibration Metrics

To measure the calibration of a network, we follow the approach as described in [9]. We initially define the number of confidence intervals M each of size $1/M$. The confidence interval I_m corresponds to the confidence range $((m-1)/M,\ m/M]$. Let B_m be the set of samples for which the predicted confidence falls in I_m. The accuracy and confidence for B_m are defined, respectively, as

$$acc(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbf{1}(\hat{y}_i = y_i)$$

$$conf(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \hat{p}_i,$$

where $\hat{y}_i, y_i,$ and $\hat{p}_i$ correspond to predicted label, true label, and the confidence (or winning score), respectively. Unless stated explicitly, confidence and accuracy will refer to a bin's confidence and accuracy.
The Expected Calibration Error (ECE) measures the amount of miscalibration in a network by computing the difference between the confidence and accuracy over the M intervals. Formally, ECE is defined as

$$ECE = \sum_{m=1}^{M} \frac{|B_m|}{n} \Big| conf(B_m) - acc(B_m) \Big|.$$

The Overconfidence Error (OE) is a weighted measure, which penalises overconfident predictions, and is defined as

$$OE = \sum_{m=1}^{M} \frac{|B_m|}{n} \left[conf(B_m) * max\left(conf(B_m) - acc(B_m),\ 0\right) \right].$$

Table 1. Fine-grain details for replicating the experiments

Property	Values	Remark
Framework	Pytorch	Version 1.3.1
Neural Networks	Resnet-18, Resnet-34, and VGG-16	–
Datasets	CIFAR-100, Fashion-MNIST, MNIST, and STL-10	More details in their corresponding results sections
Batch Normalisation	Enabled	–
Weight Initialisation	Kaiming	Assumed as the detail was not provided in the report
Batch Size	128	–
Epochs	200	–
Initial Learning Rate	0.1	Reduced by a factor of 0.5 at $60, 120, 160$ epoch
Optimiser	SGD	–
Momentum	0.9	Nesterov momentum
Weight Decay	$5\text{x}10^{-4}$	–
Number of Interval Bins	100	Communicated via email by the authors
Data Pre-processing	Enabled	Random cropping(padding=4), rotation $\pm 15^{\circ}$, horizontal flipping, standardisation

3 Implementation Details

We have followed, wherever applicable, the implementation details of authors submission. Though, some of the fine grain details on training a network and generating the reported results were lacking, for this we have made reasonable assumptions and listed below a full table of implementation details, which we followed. The details are provided in Table 1.

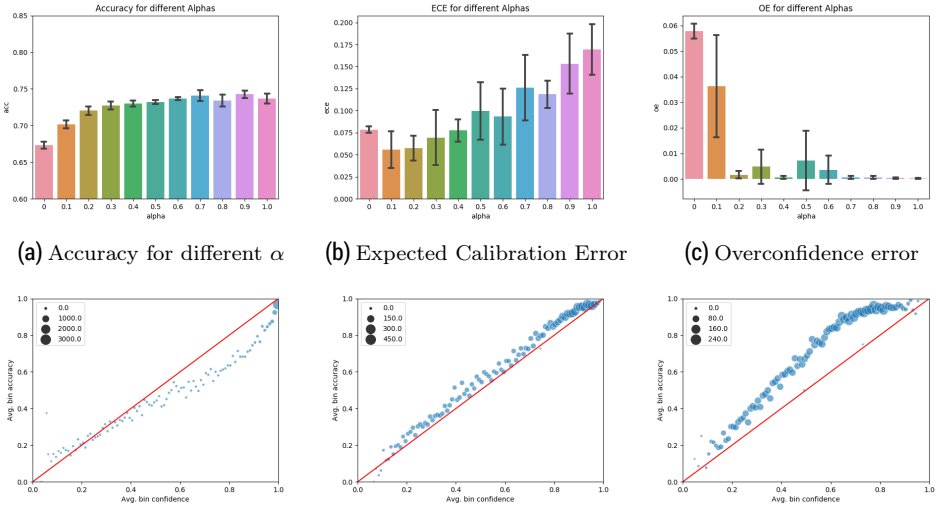
4 Replication Experiments

We conduct 4 trials per experiment, and report the mean and standard deviation over the trials. We report results for more than few values of α as opposed to the paper. The authors report the best working value of $\alpha \in [0.2, 0.4]$ hence, we use $\alpha = 0.3$ when selecting an individual value for α. Accuracy, ECE, and OE are computed for $\alpha \in [0, 1]$. Important thing to note is $\alpha = 0$ corresponds to no Mixup, which is one of the author's baseline, and $\alpha = 1$ is simple averaging of two input images. Including $\pm$ standard deviation also provides additional insights to the range of values for accuracy, ECE and OE which are, instead, missing in the paper. We also show scatter plots for $\alpha = \{0, 0.3, 1.0\}$ to compare average bin accuracy against average bin confidence. To provide additional details of the frequency of samples in the bin, we scale the point size on the plot to correspond to the size of the bin $|B_m|$.

4.1 CIFAR-100

CIFAR-100 consists of 100 classes with input images of dimensions 32x32x3. We use the standard split of the dataset consisting of $50,000$ and $10,000$ as training and test images, respectively. We train the Resnet-34 model following the details provided in the Section 3.

The results can be viewed in Figure 2. Though we were not able to obtain the accuracy value 80%, we were able to replicate the trend that the accuracy is higher when the network is trained with Mixup. For ECE and OE there are significant fluctuations in the values. The results reported by the authors lie within one standard deviation due to large value of standard deviation. However, ECE decreases till $\alpha = 0.3$ and then increases, which is also reported in paper. Figure 2 (d–f) represents the shift from the network being overconfident ($\alpha = 0$) to under confident ($\alpha = 1$). The network is better calibrated, if the points in the scatter plot lie close to the line $y = x$, and it is observed with $\alpha = 0.3$ in Figure 2 (e).



(a) Accuracy for different α (b) Expected Calibration Error (c) Overconfidence error

(d) No Mixup ($\alpha = 0$) (e) $\alpha = 0.3$ (f) Simple average ($\alpha = 1$)

Figure 2. Replication of the results corresponding to CIFAR-100 dataset.

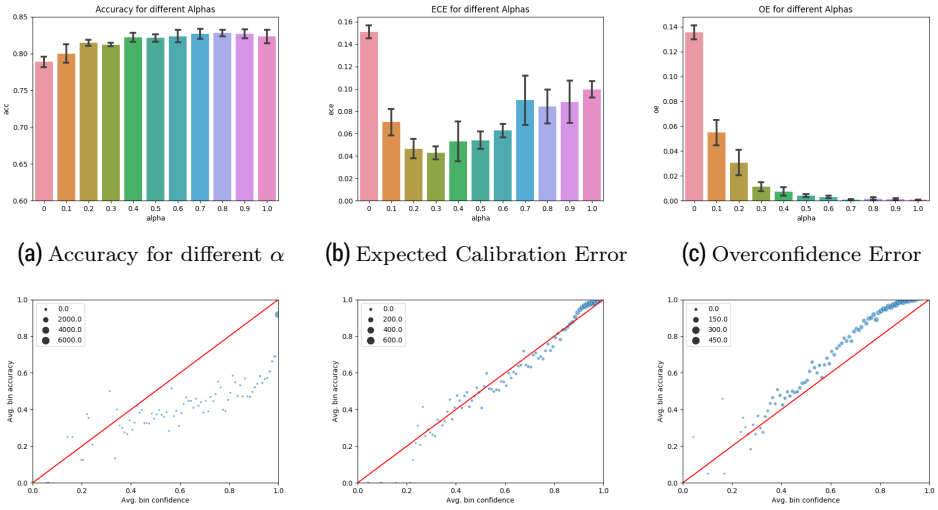
4.2 STL-10

The dataset consists of labelled as well as unlabelled examples. For the purpose of this experiment, we utilise the labelled split of train and test. The dataset consists of 10 classes with input images of dimensions 96x96x3. We train the VGG-16 model following the details provided in the Section 3.

The accuracy matches with the values reported by authors as observed in Figure 3. The trend of improved accuracy due to Mixup is also observed. For ECE, and OE the values for no Mixup were off, but the expected trend of improved calibration was observed. Again, $\alpha \in [0.2, 0.4]$ can be seen as the best value w.r.t ECE. Figure 3 (d–f) represents the shift from the network being overconfident ($\alpha = 0$) to under confident ($\alpha = 1$).

4.3 Fashion-MNIST

The dataset consists of a training set of $60,000$ examples and a test set of $10,000$ examples. Each example is a 28x28x1 gray-scale image, associated with a label from 10 classes. We use the standard splits for training and testing.

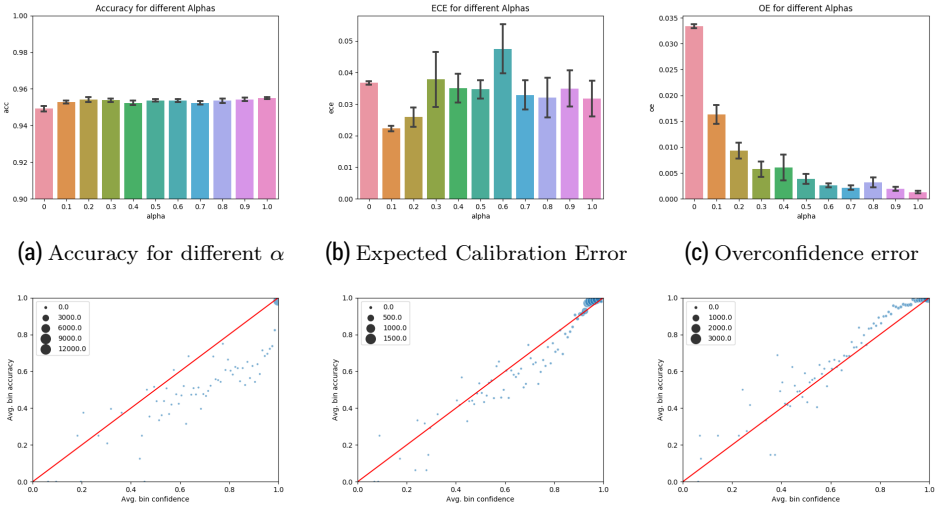


(a) Accuracy for different α (b) Expected Calibration Error (c) Overconfidence Error

(d) No Mixup ($\alpha = 0$) (e) $\alpha = 0.3$ (f) Simple average ($\alpha = 1$)

Figure 3. Replication of the results for STL-10 dataset.

The effect of Mixup on accuracy are not quite evident, since the network learns to classify effectively even without Mixup, Figure 4. ECE value is lower for majority of α indicating better calibration. The scatter-plot is not a good indicator to analyse the calibration in this case as majority of the samples are easily classified by the neural network with high accuracy. Only OE behaves a expected suggesting that Mixup helps in calibrating the neural network to avoid over confident predictions.



(a) Accuracy for different α (b) Expected Calibration Error (c) Overconfidence error

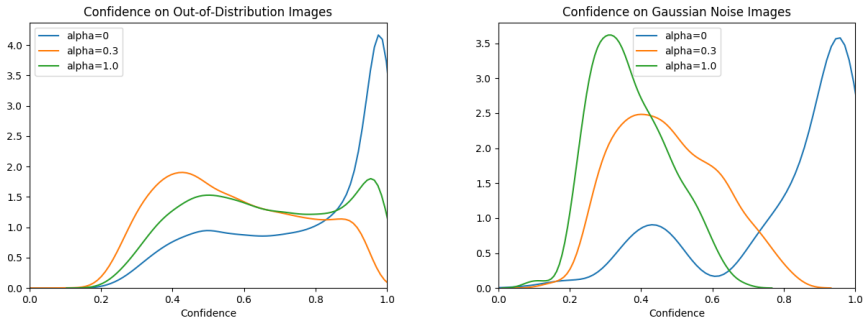
(d) No Mixup ($\alpha = 1$) (e) $\alpha = 0.3$ (f) Simple average ($\alpha = 1$)

Figure 4. Replication of the results for Fashion-MNIST dataset.

4.4 Testing on Out-Of-Distribution dataset & Random Noise

In the original work, authors have trained a neural network on STL-10 (in-distribution data) and tested on the samples from ImageNet (out-of-distribution data) [10], corresponding to new classes only. On the other hand, we aim to replicate the effectiveness of

this experiment by training on Fashion-MNIST (in distribution data) and test on MNIST (out-of-distribution data). In principle, the findings of our experiment should go along with the reported results.
MNIST, similar to Fashion-MNIST, consists of $50,000$ training and $10,000$ test images of dimensions 28x28x3. We train the Resnet-18 network on the train split of Fashion-MNIST and report the results on the test split of MNIST. For random noise images, we use Gaussian noise with mean and standard deviation of the Fashion-MNIST dataset to generate 1024 test images.
In Figure 5, the plots closely resemble to the outputs of the paper. For both the scenarios, we can observe that the network trained with Mixup is significantly less confident in predicting the test samples. For random noise, the predictions are somewhat separable as reported by the authors.



(a) Out-of-Distribution data (b) Gaussian noise data

Figure 5. Distribution of confidence of various models when tested on out-of-distribution and Gaussian noise samples, after being trained on the Fashion-MNIST dataset

5 Conclusion

In this report, we were able to confirm the main results reported in [1] by replicating the majority of the experiments. Though the values were not exactly replicated for ECE and OE, the trend of improved calibration when the network is trained using Mixup was confirmed. We were able to obtain the authors finding regarding the value of $\alpha \in [0.2, 0.4]$, providing the best calibration in terms of ECE. Through the out-of-distribution and random noise experiments in Section 4.4, we were able to provide some evidence regarding the generalisability of the author's corresponding experiment, thereby strengthening their claim of improved calibration on unseen data. The decrease of OE with increase in α is also confirmed, which the authors attributed to underfitting of the model. An important finding from the replication of the experiments was the volatility of the error values, hence we urge future researchers to report mean and variance for the errors to allow for better comparisons. We also provide our source code [1] for generating the results and setting up the experiments.

References

1. S. Thulasidasan, G. Chennupati, J. A. Bilmes, T. Bhattacharya, and S. E. Michalak. "On Mixup Training: Improved Calibration and Predictive Uncertainty for Deep Neural Networks." In: **ArXiv** abs/1905.11001 (2019).
2. H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. "mixup: Beyond Empirical Risk Minimization." In: **International Conference on Learning Representations** (2018).

[1] https://github.com/MacroMayhem/OnMixup

3. A. Krizhevsky. **Learning multiple layers of features from tiny images**. Tech. rep. 2009.
4. H. Xiao, K. Rasul, and R. Vollgraf. **Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms**. Aug. 28, 2017. arXiv: cs.LG/1708.07747 [cs.LG].
5. A. Coates, A. Ng, and H. Lee. "An Analysis of Single-Layer Networks in Unsupervised Feature Learning." In: **Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics**. Ed. by G. Gordon, D. Dunson, and M. Dudík. Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, Nov. 2011, pp. 215–223.
6. V. Vapnik. "Principles of Risk Minimization for Learning Theory." In: **Advances in Neural Information Processing Systems 4**. Ed. by J. E. Moody, S. J. Hanson, and R. P. Lippmann. Morgan-Kaufmann, 1992, pp. 831–838.
7. V. N. Vapnik. **The Nature of Statistical Learning Theory**. Berlin, Heidelberg: Springer-Verlag, 1995.
8. O. Chapelle, J. Weston, L. Bottou, and V. Vapnik. "Vicinal Risk Minimization." In: **Advances in Neural Information Processing Systems 13**. Ed. by T. K. Leen, T. G. Dietterich, and V. Tresp. MIT Press, 2001, pp. 416–422.
9. C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. "On Calibration of Modern Neural Networks." In: **Proceedings of the 34th International Conference on Machine Learning - Volume 70**. ICML'17. Sydney, NSW, Australia: JMLR.org, 2017, pp. 1321–1330.
10. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. "ImageNet: A Large-Scale Hierarchical Image Database." In: **CVPR09**. 2009.