

Plant Disease Prediction Using CNN-IDML

A Deep Learning Approach for Cassava Leaf Disease Classification

Suvramoy Pal & Chandresh Joshi
(Roll Number: 232123028 & 232123007)

Under the supervision of
Prof. Rajen Kumar Sinha
Department of Mathematics
Indian Institute of Technology Guwahati
Guwahati - 781039, India

May 2025

Overview

1. Introduction
2. Objective of the Project
3. Dataset Description
4. Methodology
5. Neural Networks for Plant Disease Detection
6. CNN Results
7. Introspective Deep Metric Learning
8. Conclusion

INTRODUCTION

Importance of Plant Disease Prediction

- **Agricultural Impact:**
 - Diseases reduce crop yields by up to 30%, causing financial losses and food insecurity.
- **Challenges of Conventional Methods:**
 - Time-consuming and resource-intensive.
 - Misses early-stage diseases not visible to the naked eye.

Our Approach

Convolutional Neural Networks (CNNs):

- Extracts features (edges, textures, disease patterns) from raw images.
- Uses ResNet-50 pretrained on ImageNet.

Introspective Deep Metric Learning (IDML):

- Generates semantic and uncertainty embeddings.
- Uses Proxy-Anchor Loss for robust clustering.

Goal

Develop a reliable system to distinguish similar disease symptoms and handle real-world image variability.

Objective of the Project

- To detect plant diseases early and accurately.
- To use CNN and improve performance with IDML

Dataset Description

- **Source of dataset:** Kaggle
- **Number of images:** 21,397
- **Number of classes (healthy + diseases):** 5

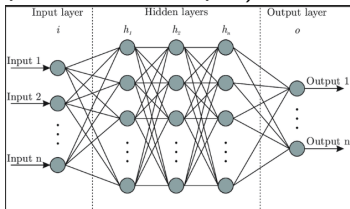
Sample Images from Each Class:



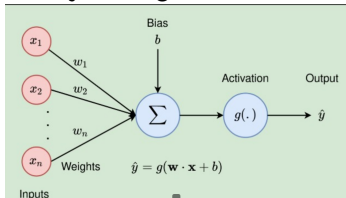
- **Preprocessing steps:**
 - **Data Augmentation:**
 - Rotation: Up to 20 degrees for varied viewpoints.
 - Width/Height Shifts: 20% for positional variations.
 - Zoom: 20% to simulate different scales.
 - Horizontal Flips: 50% probability to reflect mirrored perspectives.
 - **Resizing Images:** All input images were resized to a fixed dimension (224×224 pixels) to ensure uniformity across the dataset and to reduce computational cost.
 - **Normalization:** Pixel values were normalized from $[0, 255]$ to $[0, 1]$ by dividing by 255, ensuring consistent inputs.

Artificial Neural Networks (ANNs)

- **Structure:** Layers of nodes (input, hidden, output) mimicking human brain processing.

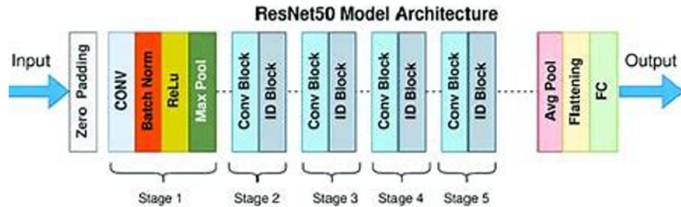


- **Operation:**
 - Neurons scale inputs by weights, sum them, and apply activation functions (e.g., ReLU, sigmoid).
 - Trained via backpropagation to adjust weights.

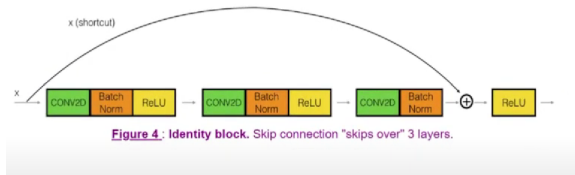
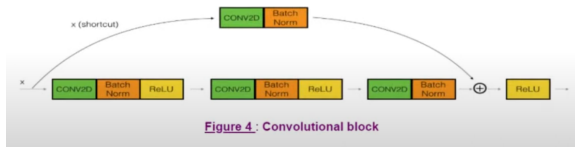


Why Convolutional Neural Networks?

- **Designed for Images:**
 - Preserves spatial structure of pixel grids.
 - Detects local patterns (e.g., leaf edges, disease spots).
- **Advantages:**
 - **Local Pattern Recognition:** Filters detect features while tracking their positions.
 - **Efficiency:** Shared weights reduce computational load.
 - **Hierarchical Learning:** From basic edges to complex disease patterns.
 - **Automated Feature Extraction:** No manual feature engineering required.
- **Suitability:** Ideal for distinguishing subtle disease symptoms in varied images.



CNN Building Blocks



- **Convolutional Layer:**

- Applies filters to detect features (low level/high level fetures e.g. edges etc).
- Output size: $W_{\text{out}} = \lfloor (n - f + 2p)/s \rfloor + 1$.
- Introduces non-linearity (e.g., ReLU: $f(x) = \max(0, x)$).
- Softmax for class probabilities.

CNN Building Blocks (Cont.)

- **Pooling Layer:**
 - Reduces feature map size (max pooling retains maximum value) keeping the most important parts and discarding the rest.
 - Primarily done to reduce overfitting.
 - Pooling makes features location-independent (translation invariance)
 - Output size: $W_{\text{out}} = \lfloor (n - f)/s \rfloor + 1$.
- **Padding:** Adds zeros to preserve border information.
- **Stride:** Stride refers to how many pixels the filter moves (or "slides") across the input image at each step during convolution.
- **Fully Connected Layer:**
 - Aggregates features for final disease classification.
 - Enhanced by IDML for better clustering.
- **Integration:** Layers work together to detect and classify disease patterns robustly.

Gradient Analysis

- **Categorical Cross-Entropy Loss:**

$$\mathcal{L}(y, \hat{y}) = - \sum_{i=1}^C y_i \log(\hat{y}_i)$$

- Encourages the model to assign high probability to the correct class.
- Only the true class term ($y_i = 1$) contributes to the loss.
- **Gradient of Loss w.r.t Logits (Softmax + Cross-Entropy):**

$$\frac{\partial \mathcal{L}}{\partial z_j} = \hat{y}_j - y_j$$

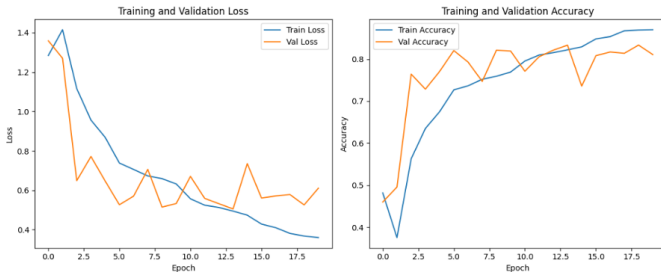
- For misclassified points, this gradient is non-zero.
- **Backpropagation Weight Update Rule:**

$$W_{\text{new}} = W_{\text{old}} - \eta \cdot \frac{\partial \mathcal{L}}{\partial W}$$

- Large gradients (from high loss) lead to bigger weight changes

CNN Results

- Trained with Cross-Entropy Loss, the baseline ResNet-50 model achieves a commendable validation accuracy of **79.30%**, reflecting good performance in cassava leaf disease classification.
- Evaluation Metrics:
 - **Precision:** 75.40% — The proportion of predicted positives that are actually correct.
 - **Recall:** 79.45% — The proportion of actual positives that were correctly identified.
 - **F1-score:** 80.04% — The harmonic mean of precision and recall, balancing both.



Introduction to Deep Metric Learning

- **Motivation of an Uncertainty-Aware Metric:** Let X be an image set with N training samples $\{x_1, \dots, x_N\}$ and L be the ground truth label set $\{l_1, \dots, l_N\}$. Deep metric learning methods aim at learning a mapping to transform each image x_i to an embedding space, where conventional methods use a deterministic vector embedding y_i to represent an image. They usually adopt the Euclidean distance as the distance measure

$$D(x_1, x_2) = D_E(x_1, x_2) = \|y_1 - y_2\|_2,$$

where $\|\cdot\|_2$ denotes the L2-norm. They then impose various constraints on the pairwise distances, which generally enlarge inter-class distances and reduce intra-class distances.

Motivation of the proposed IDML framework

- **Handles Uncertainty:**

- Uncertainty naturally exists in images due to occlusion, scale, pose, low resolution, or semantic ambiguity. Most existing deep metric learning methods only consider the **semantic distance of an image pair and ignoring the possible uncertainty in the images**. Our IDML achieves this by using **both a semantic embedding and an uncertainty embedding** to describe an image and employing an introspective similarity metric to compute an uncertainty-aware distance.

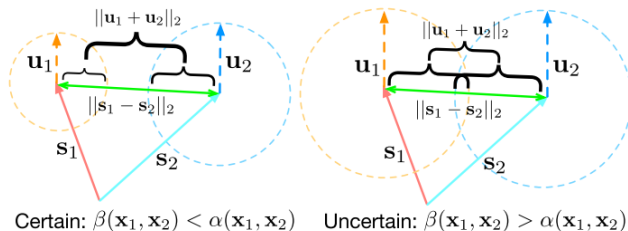
- **Advantages:**

- **Intelligent Similarity:** Combines features and uncertainty for accurate comparisons.
- **Improved Training:** Prioritizes clear images, avoids overfitting on noisy ones.
- **Robustness:** Handles real-world challenges like poor lighting or symptom overlap.

Introspective Similarity Metric:

- We represent an image using a semantic embedding s and an uncertainty embedding u , i.e., $y_{IN} = \{s, u\}$. The semantic embedding s describes the semantic characteristics of an image, while the uncertainty embedding u models the ambiguity.
- For comparing two images x_1 and x_2 , we define the **semantic distance** as $\alpha(x_1, x_2) = \|s_1 - s_2\|_2$, similar to conventional deep metric learning (DML) methods.
- We further compute a **similarity uncertainty** as $\beta(x_1, x_2) = \|u_1 + u_2\|_2$. Note that we add the vectors of the uncertainty embeddings before computing the norm, instead of directly adding their norms. The reason is that the uncertainty should depend on both concerning images. For example, it might be difficult to differentiate a wolf from a dog, but it can be affirmatively distinguished from a person.

IDML framework Conti....



- Our main aim is to determine the semantic similarity between two images, So an introspective metric need to consider both semantic and uncertainty. and when not certain enough the metric refuses to distinguish semantic differences.

- The relative uncertainty is defined as:

$$\tilde{\beta}(x_1, x_2) = \frac{\beta(x_1, x_2) + \gamma}{\alpha(x_1, x_2)} \quad (1)$$

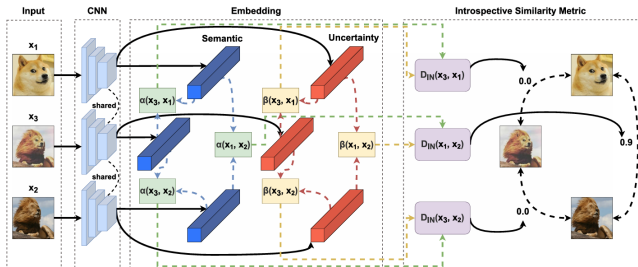
Note that the relative uncertainty is always non-negative.

- The introspective similarity metric (ISM) is computed as:

$$D_{\text{IN}}(x_1, x_2) = \alpha(x_1, x_2) \cdot \exp\left(-\frac{1}{\tau} \tilde{\beta}(x_1, x_2)\right) \quad (2)$$

where $\tau > 0$ is a hyperparameter that controls the weakening degree.

ISM flow-work(conti...)



- We employ a convolutional neural network to represent each image by a semantic embedding and an uncertainty embedding. We then use the distance between semantic embeddings as the semantic discrepancy and add the uncertainty embeddings for uncertainty measure. The introspective similarity metric then uses the uncertainty level to weaken the semantic discrepancy to make a discrete similarity judgment.

Gradient Analysis (1/2)

We provide a gradient analysis to demonstrate the effect of our introspective similarity metric on the learning of semantic embeddings. Generally, our proposed similarity metric results in reduced semantic discrepancies to lower the influence of uncertain samples. These influences are measured by the magnitude of gradients on the model parameters.

For a conventional metric learning method $J_E = J(y, L; D_E)$ with Euclidean distance $D_E(x_1, x_2) = \alpha(x_1, x_2) = \|s_1 - s_2\|_2$, the loss gradient w.r.t. model parameters W (affecting the semantic embeddings s) is:

$$\frac{\partial J_E}{\partial W} = \frac{\partial J_E}{\partial s} \cdot \frac{\partial s}{\partial W} = \frac{\partial J_E}{\partial \alpha} \cdot \frac{\partial \alpha}{\partial s} \cdot \frac{\partial s}{\partial W} \quad (1)$$

For the IDML objective $J_{IN} = J(y_{IN}, L; D_{IN})$, the loss gradient becomes:

$$\frac{\partial J_{IN}}{\partial W} = \frac{\partial J_{IN}}{\partial D_{IN}} \cdot \frac{\partial D_{IN}}{\partial s} \cdot \frac{\partial s}{\partial W} = \frac{\partial J_{IN}}{\partial D_{IN}} \cdot \frac{\partial D_{IN}}{\partial \alpha} \cdot \frac{\partial \alpha}{\partial s} \cdot \frac{\partial s}{\partial W} \quad (2)$$

Gradient Analysis (2/2)

In both (1) and (2), the term $\frac{\partial s}{\partial W}$ depends only on the architecture. Since IDML replaces D_E with D_{IN} , we get:

$$\frac{\partial J_{IN}}{\partial s} = \frac{\partial J_E}{\partial s}, \quad \text{so} \quad \frac{\partial J_{IN}}{\partial W} = \frac{\partial J_E}{\partial W} \cdot \frac{\partial D_{IN}}{\partial \alpha} = \frac{\partial J_E}{\partial W} \cdot H(\alpha, \beta)$$

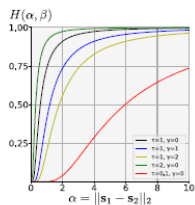
where:

$$H(\alpha, \beta) := \frac{\partial D_{IN}}{\partial \alpha} = \frac{\partial \left(\alpha \cdot \exp \left(-\frac{\beta + \gamma}{\alpha \tau} \right) \right)}{\partial \alpha} = \exp \left(-\frac{\beta + \gamma}{\alpha \tau} \right) \cdot \left(1 + \frac{\beta + \gamma}{\alpha \tau} \right)$$

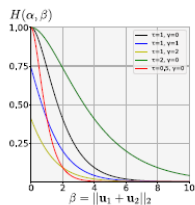
Here, $\beta^\dagger = \frac{\beta + \gamma}{\alpha}$ is the relative uncertainty. As $g(x) = e^{-x}(1+x)$ is monotonically decreasing for $x \geq 0$, $H(\alpha, \beta)$ is maximized at 1 when $\beta^\dagger = 0$ and decreases as uncertainty increases.

Thus, uncertain samples contribute smaller gradients, reducing false training signals. When $\beta^\dagger = 0$, IDML reduces to the conventional method using Euclidean distance.

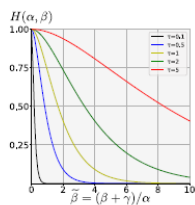
Effect of $\alpha\beta$



(a) Effect of α .



(b) Effect of β .



(c) Effect of $\tilde{\beta}$.

Diffrent losses of IDML

- There are two type of losses in IDML
- Pair-based losses where we compute the relation betwwen data to data which give rich and fined grained but demanding high traaining comlexity.
- proxy-based losses where we compute data to proxy based loss it reduce the training complexity but gives impoverished information.
- We use proxy anchor loss which is combination of rich and fine grained also impoverished information.
- The ProxyAnchor Loss, originally proposed by Kim et al.

Proxy-Anchor Loss with IDML

- **Concept:** Uses proxies to represent each disease class, reducing comparison complexity.
- **Loss Function:**

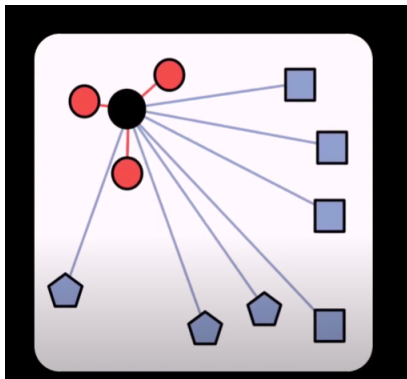
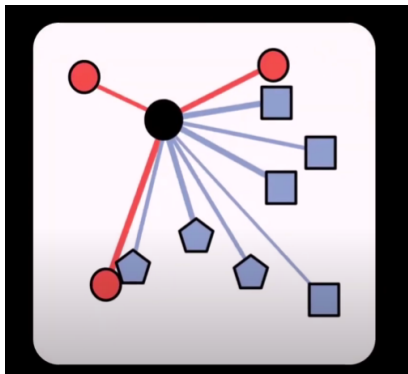
$$J_{PA} = \frac{1}{|P^+|} \sum_{p \in P^+} \left[\log \left(1 + \sum_{i: l_i = l_{p^+}} e^{-\alpha(C_{IN}(x_i, p^+) - \delta)} \right) \right] \\ + \frac{1}{|P|} \sum_{p \in P} \left[\log \left(1 + \sum_{i: l_i \neq l_p} e^{\alpha(C_{IN}(x_i, p) + \delta)} \right) \right]$$

- **Parameters Details:**
 - $\alpha = 32$: Scaling factor that controls the sensitivity of the loss function to differences between instances.
 - $\delta = 0.1$: A constant margin that shifts the distance calculations, influencing the decision boundary.
 - Embedding Size = 512 is The dimensionality of the vector representation for each input feature.

Proxy-Anchor Loss with IDML (conti...)

- $|P^+|$ is the number of positive proxies (proxies for the same class as the sample).
- $|P|$ is the total number of proxies across all classes.
- $p \in P^+$ iterates over positive proxies.
- $p \in P$ iterates over all proxies.
- $i : l_i = l_{p^+}$ indicates summation over samples x_i where the label l_i matches the label l_{p^+} of the positive proxy.
- $i : l_i \neq l_p$ indicates summation over samples x_i where the label l_i differs from the label l_p of the proxy.
- $C_{IN}(x_i, p)$ is the introspective similarity between sample x_i and proxy p .
- α is a scaling factor (set to 32 in our implementation).
- δ is the margin parameter (set to 0.1 in our implementation), enforcing separation between positive and negative pairs.

Proxy-Anchor Loss with IDML (conti...)



Pushing or pulling all embedding space in batches.

Proxy-Anchor Loss uses cosine similarity, requiring a similarity-based ISM (C_{IN}):

$$C_{IN}(x_i, p_j) = 1 - (1 - C(x_i, p_j)) \cdot \exp\left(-\frac{1}{\tau} \tilde{\beta}(x_i, p_j)\right),$$

IDML Training Algorithm (1/2)

Algorithm 1 IDML with Proxy-Anchor Loss

1. **Input:** Training dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, CNN model, proxies $\{p_j\}_{j=1}^C$, hyperparameters τ, γ, λ , learning rate η
2. Initialize model parameters θ and proxies $\{p_j\}_{j=1}^C$
3. **for** each epoch **do**
4. **for** each batch $\{(x_i, y_i)\}_{i=1}^B \subset \mathcal{D}$ **do**
5. Compute semantic embeddings $s_i = f_{\theta}^s(x_i)$ and uncertainty embeddings $u_i = f_{\theta}^u(x_i)$
6. Compute introspective similarity:

$$C_{IN}(x_i, p_j) = 1 - (1 - C(x_i, p_j)) \cdot \exp\left(-\frac{1}{\tau} \tilde{\beta}(x_i, p_j)\right)$$
7. where:

$$C(x_i, p_j) = \frac{s_i \cdot p_j}{\|s_i\|_2 \|p_j\|_2}, \quad \beta(x_i, p_j) = \|u_i + u_j\|_2, \quad \tilde{\beta}(x_i, p_j) = \frac{\beta(x_i, p_j) + \gamma}{\|s_i - p_j\|_2}$$

IDML Training Algorithm (2/2): Proxy-Anchor Loss

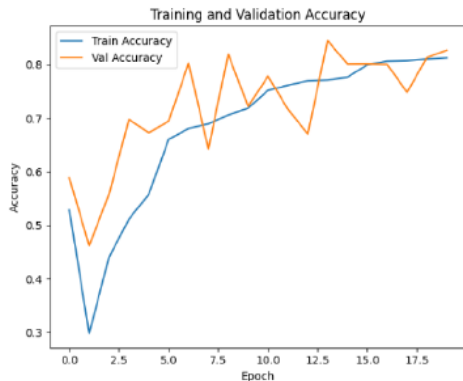
Step 8: Compute Proxy-Anchor Loss

$$\mathcal{L}_{PA} = \frac{1}{|P^+|} \sum_{p \in P^+} \log \left(1 + \sum_{i: l_i = l_{p^+}} e^{-C_{IN}(x_i, p^+)} \right) + \frac{1}{|P|} \sum_{p \in P} \log \left(1 + \sum_{i: l_i \neq l_p} e^{C_{IN}(x_i, p)} \right)$$

Training Setup

- **Parameters:**
 - Epochs: 20
 - Batch Size: 32
 - Scheduler: StepLR (reduce rate by 0.5 every 5 epochs)
 - Early Stopping: After 6 epochs with no improvement
- **Loss Functions:**
 - Baseline: Cross-Entropy Loss
- IDML: Proxy-Anchor Loss ($\alpha = 32$, margin = 0.1)
- **Hardware:** GPU for training, CPU/GPU for inference.

IDML results



Final Metrics - R01: 0.8491, R02: 0.9449, NMI: 0.4743, RP: 0.9449, MCQR: 0.9001

Evaluation Metrics

- **Primary Metric:** Classification accuracy (fraction of correctly predicted images).
- **IDML-Specific Metrics:**
 - **R@1**(recall@1): Top-1 prediction accuracy.
 - **NMI**(Normalized Mutual Information): Clustering quality via K-means (5 clusters).
 - **RP**:(Retrieval Precision): Precision of top-2 retrieval.
 - **MC@R**:(Mean class Prediction): Mean class precision for top-2 retrieval.
- **Purpose:** Evaluate both classification and retrieval performance for real-world use.

Model Performance

Model	Validation Accuracy
Baseline CNN	79.30%
IDML	84.46%

Table: Comparison of model accuracies.

IDML Metrics

- R@1: 84.91%
- NMI: 0.4743
- RP: 94.49%
- MC@R: 90.01%

Predicted Image Examples

Predicted: Healthy
True: Healthy



Predicted: Cassava Mosaic Disease (CMD)
True: Cassava Mosaic Disease (CMD)



Predicted: Cassava Mosaic Disease (CMD)
True: Cassava Brown Streak Disease (CBSD)



Predicted: Cassava Mosaic Disease (CMD)
True: Cassava Mosaic Disease (CMD)



Predicted: Cassava Mosaic Disease (CMD)
True: Cassava Green Mottle (CGM)



Challenges and Solutions






- **Class Imbalance:**
 - Solution: WeightedRandomSampler and data augmentation.
- **Image Quality Variability:**
 - Solution: IDML's uncertainty modeling flags uncertain predictions.
- **Metric Learning Complexity:**
 - Solution: Proxy-Anchor Loss optimizes embeddings efficiently.
- **Outcome:** Practical tool for farmers with limited access to technology.

Conclusion

- **Achievement:** IDML model achieves 84.46% validation accuracy, outperforming baseline CNN (79.30%).
- **Key Strengths:**
 - Robust to real-world image variability.
 - High retrieval precision (RP: 94.49%, MC@R: 90.11%).
 - Deployable on mobile devices for smallholder farmers.
- **Impact:** Enables rapid disease diagnosis, reducing crop losses

- **Enhanced Architectures:** Explore advanced CNNs or transformers for better clustering (improve NMI).
- **Expanded Dataset:** Include more diverse images to capture rare disease cases.
- **Real-World Testing:** Deploy and validate model in field conditions with farmers.
- **Integration:** Develop user-friendly mobile apps for broader accessibility.

References

-  Kaggle, "Cassava Leaf Disease Classification," 2020.
<https://www.kaggle.com/competitions/cassava-leaf-disease-classification>.
-  I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.
-  C. Wang et al., "Introspective Deep Metric Learning," *IEEE Transactions*, 2023.
<https://ieeexplore.ieee.org/abstract/document/10239539>.
-  G. Shrestha, M. Das, N. Dey, "Plant Disease Detection Using CNN," *2020 IEEE Applied Signal*, 2020.
-  A. Babenko et al., "Neural Codes for Image Retrieval," *ECCV*, 2014.

The End