Supplementary Document for GlideNet: Global, Local and Intrinsic based Dense Embedding NETwork for Multi-category Attributes Prediction

S.1 Introduction

This is a supplementary document that contains some useful information for accurately reproducing findings, as well as the reasoning for using Visual Attributes in the Wild (VAW) and Cityscapes Attributes Recognition (CAR) datasets for experiments in the GlideNet paper.

The complete specifications and setup of the proposed network – Global, Local and Intrinsic based Dense Embedding Network (GlideNet) – architecture are stored in Section S.2.

Section S.3 discusses the two datasets, VAW and CAR datasets, used in the evaluation and why we chose those two datasets in particular.

All of the configurations for training GlideNet are contained in Section S.4.

S.2 Network Architecture Details

In this section, we discuss the exact details of each building block of GlideNet. The reader is encouraged to read Section 3.1 in the main document (GlideNet-paper) first to understand the purpose of each building block. Here, we only show the configurations without a detailed description of the purpose. In our training algorithms (Section 3.2), we have two stages. The first stage uses 'temporary' decoders that are removed later in both Stage II of training and the inference stage. The details of these temporary decoders are found in Sections S.2.6 to S.2.8

S.2.1 Feature Extractors (FEs)

For all Feature Extractors (FEs), we use the backbone of ResNet-50 [3]. Specifically, we take the output after layers $2, 3, \& 4$ as our features. The inputs to the FEs are always resized to 224×224. Since the output features don't match in spatial dimensions, we upsample them to the size of the largest, which is 28×28. The total number of the output channels for each FE in this case is $128 + 128 + 512 = 768$. In the case

Table S.1: Structure of Object Descriptor ($\mathcal{D}$)

Layer Name	$\mathcal{D}$.C.1	$\mathcal{D}$.C.2	$\mathcal{D}$.M.1	$\mathcal{D}$.M.2	$\mathcal{D}$.P.1	$\mathcal{D}$.P.2
Input	$\hat{\mathbf{c}}$	$\mathcal{D}$.C.1	M	$\mathcal{D}$.M.1	$\mathcal{D}$.M.2 $\bigotimes$ $\mathcal{D}$.C.2	$\mathcal{D}$.P.1
Structure	[Linear(2260, 512) relu]	[Linear(512, 32) softmax]	[3×3 conv$(1, 16)/2$ batch norm relu]	[3×3 conv$(16, 32)/2$ batch norm relu]	[3×3 conv$(32, 64)$ batch norm relu upsample(2)]	[3×3 conv$(64, 128)$ batch norm sigmoid upsample(2)]
Output	512	32	$16 \times 112 \times 112$	$32 \times 56 \times 56$	$64 \times 112 \times 112$	$128 \times 224 \times 224$ D

of Instance Feature Extractor (IFE), we replace each convolution layer with the novel layer proposed in Section 3.3 of the main document. However, other than the usage of the (Mask-)Informed Convolution concept, it has the same exact structure as other FEs. The output of Global Feature Extractor (GFE), Local Feature Extractor (LFE) and IFE is denoted by F_G, F_L and F_I respectively.

We initialized the weights of the FEs with the pretrained model found in PyTorch framework [6] that is initially trained for classification problem with the ImageNet dataset [1]. For IFE, we initialize the weights of all Informed Convolution layers with their corresponding 'normal' convolution layers found in the pretrained model. The reason is that a 'normal' convolution layer is a special case of the Informed Convolution layer and it can be a good initialization for the weights of IFE. However, we found that the difference in performance is insignificant between with and without the pretrained initialization of the IFE.

S.2.2 Object Descriptor

The Object Descriptor, Table S.1, is depicted in Fig. 3 in the paper. It has two input branches for the category embedding and the binary mask of the object. The category embedding branch $\mathcal{D}.C$ consists of two fully connected layers while the binary mask branch $\mathcal{D}.M$ consists of two 2D-Convolution layers. The output of $\mathcal{D}.C$ is broadcasted and multiplied with $\mathcal{D}.M$ as in Eq. (5).

We have investigated the usage of the cropped image I_C in the generation of the description D. However, we noticed that it actually increases the complexity while not increasing the performance. Since all important features are extracted from $I\&I_C$ by our strong FEs. The object descriptor needs only to 'learn' where to give attention. This can mainly be obtained by information about 1) where the object is located in the input image I_C which is provided through the binary mask M, and 2) what category this object belongs to which is provided by the self-learned category embedding $\hat{\mathbf{c}}$.

S.2.3 Gating Mechanism

We have three gates $\mathcal{G}_G, \mathcal{G}_L\&\mathcal{G}_I$ for the three different extracted features $F_G, F_L\&F_I$ and their corresponding outputs are $A_G, A_L\&A_I$, respectively. Table S.2 shows the architecture of each gate. The input is D (the description of the object), which is the output of $\mathcal{D}$ (the Object Descriptor). We use a final Sigmoid activation function to ensure that the output range of the learned attention is between 0 and 1. This guarantees the numerical stability of the network, as later the produced values $A_G, A_L\&A_I$ are

Table S.2: Structure of Gates ($\mathcal{G}$)

Layer Name	$\mathcal{G}.1$	$\mathcal{G}.2$	$\mathcal{G}.3$
Input	D	$\mathcal{G}.1$	$\mathcal{G}.2$
Structure	$\begin{bmatrix} 3 \times 3 \text{ conv}(128, 64)/2 \\ \text{batch norm} \\ \text{relu} \end{bmatrix}$	$\begin{bmatrix} 3 \times 3 \text{ conv}(64, 32)/2 \\ \text{batch norm} \\ \text{relu} \end{bmatrix}$	$\begin{bmatrix} 3 \times 3 \text{ conv}(32, 3)/2 \\ \text{batch norm} \\ \text{sigmoid} \end{bmatrix}$
Output	$64 \times 112 \times 112$	$32 \times 56 \times 56$	$3 \times 28 \times 28$ A_G, A_L or A_I

Table S.3: Structure of Interpreter ($\mathcal{I}$)

Layer Name	$\mathcal{I}$.E.1	$\mathcal{I}$.E.2	$\mathcal{I}.\text{H}_i.1$	$\mathcal{I}.\text{H}_i.2$
Input	f_T	$\mathcal{I}$.E.1	$\mathcal{I}$.E.2	$\mathcal{I}.\text{H}_i.1$
Structure	$\begin{bmatrix} \text{Linear}(768, 512) \\ \text{relu} \end{bmatrix}$	$\begin{bmatrix} \text{Linear}(512, 256) \\ \text{softmax} \end{bmatrix}$	$\begin{bmatrix} \text{Linear}(256, 128) \\ \text{relu} \end{bmatrix}$	$\begin{bmatrix} \text{Linear}(128, 620) \end{bmatrix}$
Output	512	256	128	620 $\hat{\mathbf{a}}$

multiplied with the features $F_G, F_L \& F_I$. Without bounding the range of the learned attention maps, the values may explode.

S.2.4 Interpreter

The interpreter, Table S.3, consists of two stages as shown in Fig. 4 of the main document. The first part $\mathcal{I}.E$ reduces the length of the feature vector to 256. Then for each category, we have its own $\mathcal{I}.\text{H}_i, \forall i \in \{1, 2, \cdots c\}$, where c is the number of categories. The output length in the case of VAW is 620 as the set of attributes is fixed over all categories. However, the output length in the case of CAR varies depending on the set of possible attributes of each category. For example, the output length for a Pedestrian object would be 38. While the output length for a Mid-to-Large Vehicle is 41. Please refer to Sections S.3 and 4 for more details and discussion about the datasets.

S.2.5 Category Estimator

The first column of Table S.4 shows the structure of the category estimator, which is a single fully connected layer with an output length equal to the number of categories.

Table S.4: Structure of Category Estimator ($\mathcal{C}$) and local and intrinsic attributes estimators (LAE & IAE)

Layer Name	$\mathcal{C}.1$	LAE.1	IAE.1
Input	F_L	F_L	F_I
Structure	$\begin{bmatrix} \text{Linear}(768, 2260) \end{bmatrix}$	$\begin{bmatrix} \text{Linear}(768, 620) \end{bmatrix}$	$\begin{bmatrix} \text{Linear}(768, 620) \end{bmatrix}$
Output	2260 $\hat{\mathbf{c}}$	620 $\hat{\mathbf{a}}_l$	620 $\hat{\mathbf{a}}_i$

Table S.5: Structure of Multi-Object Detection Head (MODH)

Layer Name	MODH.1	MODH.2	MODH.3
Input	F_G	MODH.1	MODH.2
Structure	$\begin{bmatrix} 3 \times 3 \text{ conv}(768, 512) \\ \text{batch norm} \\ \text{relu} \end{bmatrix}$	$\begin{bmatrix} 3 \times 3 \text{ conv}(512, 256) \\ \text{batch norm} \\ \text{relu} \end{bmatrix}$	$\begin{bmatrix} 3 \times 3 \text{ conv}(256, 2265) \\ \text{batch norm} \\ \gamma \end{bmatrix}$
Output	$512 \times 28 \times 28$	$256 \times 28 \times 28$	$2265 \times 28 \times 28$ $\hat{O}_G$

Table S.6: Structure of Mask Estimator ($\mathcal{M}$)

Layer Name	$\mathcal{M}.1$	$\mathcal{M}.2$	$\mathcal{M}.3$	$\mathcal{M}.4$
Input	F_L	$\mathcal{M}.1$	$\mathcal{M}.2$	$\mathcal{M}.3$
Structure	$\begin{bmatrix} 3 \times 3 \text{ conv}(768, 256) \\ \text{batch norm} \\ \text{relu} \\ \text{upsample}(2) \end{bmatrix}$	$\begin{bmatrix} 3 \times 3 \text{ conv}(256, 128) \\ \text{batch norm} \\ \text{relu} \\ \text{upsample}(2) \end{bmatrix}$	$\begin{bmatrix} 3 \times 3 \text{ conv}(128, 64) \\ \text{batch norm} \\ \text{relu} \\ \text{upsample}(2) \end{bmatrix}$	$\begin{bmatrix} 3 \times 3 \text{ conv}(64, 1) \\ \text{batch norm} \\ \text{sigmoid} \end{bmatrix}$
Output	$256 \times 56 \times 56$	$128 \times 112 \times 112$	$64 \times 224 \times 224$	$1 \times 224 \times 224$ $\hat{M}$

Ideally, the Category Estimator should produce a one-hot encoding representation of the category of the object. However, practically it produces a Probability Mass Function (PMF) of the object's category. If the PMF has two or more peaks, then that is primarily due to the visual similarity between their corresponding categories. In other words, the generated embedding captures visual similarities between different categories based on the shape of the object of interest. As we have argued in Table 5, this is better than a fixed pretrained word embedding such as GloVe [7] in Pham *et al*. [8].

S.2.6 Multi-Object Detection Head

The multi-object-detection head (MODH), Table S.5, detects different objects in the whole given image. The output should not change by using different objects in the same image as the input of GFE is the whole input image I. The input to the MODH is the upsampled and concatenated features (F_G) Eq. (1). The output has 2265 channels in case of using VAW dataset and 17 in case of using CAR, since the number of categories is different in each dataset. The final output passes by a custom activation function γ that splits the channels of the input features into sets and passes each set to a different activation function depending on the purpose of this set. γ is defined as follows. For the first channel that represents the confidence, the activation is a Sigmoid as well as for the bounding box center coordinates. On the other hand, the activation is an exponential for the dimensions of the bounding box (to ensure non-negativity) while it is a softmax for the multi-category channels (to ensure PMF axioms – non-negativity and summation to one).

S.2.7 Mask Estimator

The Mask Estimator ($\mathcal{M}$) is a temporary decoder that takes the output features of LFE (F_L) Eq. (2). Its structure is depicted in Table S.6. The output is a single channel representing the estimation of the binary mask. The output is restricted to be between 0 and 1 through a final Sigmoid layer – one indicates the pixel belongs to the object while zero indicates that this pixel does not belong to the object of interest.

S.2.8 Attribute Predictors

The second and third column of Table S.4 shows the structure of the local and intrinsic temporary decoders respectively. They both have the same structure, however, the inputs and outputs of each are distinct. In the case of LAE, the inputs and outputs are F_L Eq. (2) and $\hat{\mathbf{a}}_\mathbf{l}$ respectively. While for IAE, the inputs and outputs are F_I Eq. (3) and $\hat{\mathbf{a}}_\mathbf{i}$ respectively.

S.3 Discussion of Datasets

In Section 4, we have trained GlideNet using two new and challenging datasets VAW [8] and CAR [5]. The following are a few of the reasons why these two datasets were chosen in particular:

1. the number and diversity of categories in both datasets are high enough to evaluate the validity and effectiveness of the proposed multi-category architecture.
2. VAW has already been used in attributes prediction as in [8].
3. CAR has a complex taxonomy with different sets of attributes depending on different categories. Furthermore, unlike VAW, the attributes in CAR are not binary or ternary. This makes CAR more challenging as well as adds flavor to the comparison of GlideNet with other methods.

Figs. S.1 and S.2 show examples from the original papers of both datasets. CAR[1] focuses on the application of attributes prediction for self-driving vehicles. Therefore, CAR focus on attributes such as the activity of a pedestrian, visibility of a vehicle, the color of a traffic light, the speed limit of a traffic sign. On the other hand, VAW[2] is pretty generic and has a wider variety of categories but it has the same set of attributes for all. Most of those attributes are unlabeled; since the majority of them are not meaningful to a certain category. All attributes in VAW can take one of three different labels; positive, negative, and unlabeled.

S.3.1 Objects with low pixel-count:

Fig. S.3 depicts two examples that demonstrate the importance of IFE. Let's look at the first example (a), (b) & (c), a narrow vertical pole. To predict its attributes without

[1] An API is provided at https://github.com/kareem-metwaly/CAR-API
[2] The authors provide the dataset through their website http://vawdataset.com/

Figure S.1: Examples of the CAR Dataset. Figure from the original paper [5].

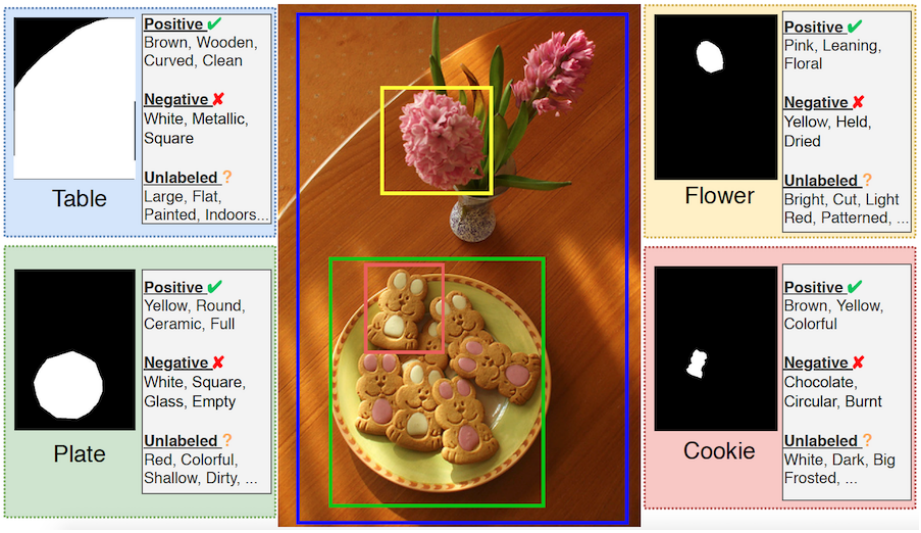

Figure S.2: Examples of the VAW Dataset. Figure from the original paper [8]

(a) The pole is the object of interest

(b) Cropped

(c) Stretched

(d) The floor is the object of interest

Figure S.3: Objects that have benefited from IFE

using IFE, we can either crop while keeping the aspect ratio (no distortion) or stretch it (distorting the image). And both techniques influence the prediction of other attributes. For example, image (c) no longer looks like a pole. Additionally, (d) displays another example (from the VAW dataset) where cropping and stretching are difficult due to the toilet being surrounded by the floor. IFE would easily help these cases, allowing information to flow only from pixels of interest.

S.4 Experimental Setup

In our Experiments and Results Section 4, we have compared GlideNet with four different state-of-the-art methods [2, 4, 9, 8]. We also performed ablation study to prove the importance and effectiveness of the different FEs as well as the novel convolution layer – Informed Convolution.

In our training, we have trained the network at Stage I for 15 epochs, then we switched to Stage II for 10 epochs. We have noticed that changing the number of

epochs slightly for each stage did not have a noticeable change in performance. The temporary decoders are removed during Stage II and inference stage; they are only used in Stage I to guide the FEs for their supposed objectives. As mentioned in Section 4, we set the values of the hyperparameters of the training loss function to be as follow: $\lambda_{gp0} = 1$, $\lambda_{gp} = 0.01$, $\lambda_{gd} = 0.5$, $\lambda_{gc} = 0.5$, $\lambda_{lm} = 0.1$, $\lambda_{lc} = 0.01$, $\lambda_{la} = 1$, $\lambda_{ia} = 1$ and $\lambda_{lc2} = 0.01$. We have attempted training with different values. What we noticed the most is that it is important to keep the values of $\lambda_{gp}, \lambda_{lc} \& \lambda_{lc2}$ lower than other values significantly. Otherwise the training diverges and the training focuses more on estimating the correct category than actually ensuring decent performance for other decoders.

We have developed our code based on the PyTorch framework [6]. We used GPUs to speed up the training, specifically, we used NVIDIA Tesla V100 ®GPUs. We distributed the code over 4 GPUs to speed up the training. The training time was less than one day and the average inference time per an entire image was ~ 0.05 second. This is a very reasonable time for a real-time application. Typically, it is not required to predict attributes with a frequency greater than 20 Hz in most applications such as autonomous vehicles. For an average speed of 60 MPH (88 feet/second), a vehicle will on average predict attributes of the scene every 4 to 5 feet $(88/20)$. This is a small traveling distance for the scene to change significantly. In other words, 20 Hz is sufficient to estimate attributes of all objects in a scene and make fast real-time decisions based on the predicted attributes. It is worth noting that this time can significantly be reduced per scene if we kept the produced General Features F_G from one instance to another in the same scene (image); as they all share the same scene. In addition, the time can be reduced by tracking objects with predicted attributes. This will help in decreasing the number of objects that require attributes prediction in each scene, which in turn decreases the computation time per scene.

References

[1] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 2

[2] Thibaut Durand, Nazanin Mehrasa, and Greg Mori. Learning a Deep ConvNet for Multi-Label Classification With Partial Labels. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 647–657, Long Beach, CA, USA, June 2019. IEEE. 7

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1

[4] Huaizu Jiang, Ishan Misra, Marcus Rohrbach, Erik Learned-Miller, and Xinlei Chen. In defense of grid features for visual question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10267–10276, 2020. 7

[5] Kareem Metwaly, Aerin Kim, Elliot Branson, and Vishal Monga. CAR – cityscapes attributes recognition a multi-category attributes dataset for autonomous vehicles. https://arxiv.org/abs/2111.08243, 2021. 5, 6

[6] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative

style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019. 2, 8

[7] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, Oct. 2014. Association for Computational Linguistics. 4

[8] Khoi Pham, Kushal Kafle, Zhe Lin, Zhihong Ding, Scott Cohen, Quan Tran, and Abhinav Shrivastava. Learning to predict visual attributes in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13018–13028, 2021. 4, 5, 7

[9] Nikolaos Sarafianos, Xiang Xu, and Ioannis A Kakadiaris. Deep imbalanced attribute classification using visual attention aggregation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 680–697, 2018. 7