Adversarial Focal Loss: Asking Your Discriminator for Hard Examples

Fox Scientist, **Panda Scientist**, **Iceberg Scientist**, **Bear Scientist**, and **Boss Scientist**

Anonymized to facilitate double-blind review.

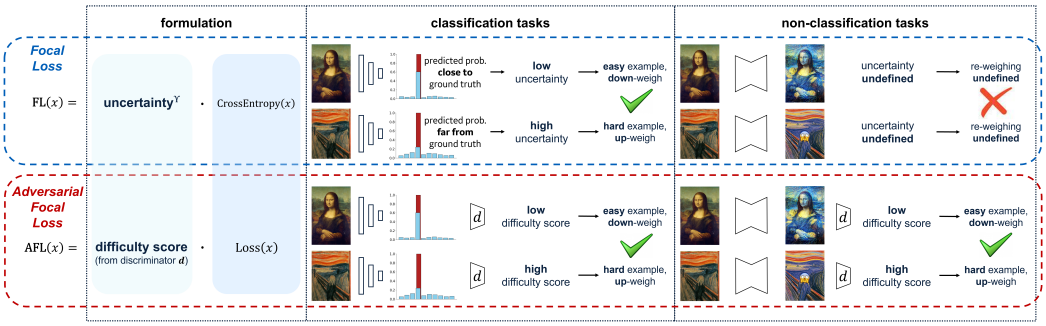

Figure 1: **Comparison between Focal Loss (FL) and the proposed Adversarial Focal Loss (AFL).** Both utilize a multiplicative coefficient to re-weigh examples based on their uncertainty/difficulty. However, AFL is more generalizable compared to Focal Loss. (1) AFL can be integrated into a variety of base loss functions (e.g., L1 loss, L2 loss, dice loss, keypoint accuracy loss, etc.) in addition to the Cross Entropy loss in the original formulation of Focal Loss. (2) Unlike Focal Loss, since AFL does not rely on the uncertainty from a classifier, it can be useful in tasks besides classification. Details about the formulations of FL and AFL can be found in section 2 and section 3, respectively.

Abstract

Focal Loss has reached incredible popularity as it uses a simple technique to identify and utilize hard examples to achieve better performance on classification. However, this method does not easily generalize outside of classification tasks, such as in keypoint detection. In this paper, we propose a novel adaptation of Focal Loss for keypoint detection tasks, called Adversarial Focal Loss (AFL). AFL not only is semantically analogous to Focal loss, but also works as a plug-and-chug upgrade for arbitrary loss functions. While Focal Loss requires output from a classifier, AFL leverages a separate adversarial network to produce a difficulty score for each input. This difficulty score can then be used to dynamically prioritize learning on hard examples, even in absence of a classifier. In this work, we show AFL's effectiveness in enhancing existing methods in keypoint detection and verify its capability to re-weigh examples based on difficulty.

1 Introduction

Machine learning, similar to human learning, is trying to draw high-level knowledge from accessible data. Just as no two leaves are alike, the values of different data samples may also vary. Surprisingly, in machine learning it's not uncommon to turn a blind eye to such intrinsic differences in data, since it is often unclear how we shall treat data differently and doing so may be prohibitively labor intensive.

Preprint. Under review.

Arguably the most obvious form of the intrinsic disparity among data samples is the so-called *class imbalance* [1, 2, 3] problem. It refers to the scenario where data samples of different classes are presented with significantly unequal frequencies. If the problem is left unattended, the resulting machine learning models can get heavily biased towards predicting majority classes.

Fortunately, this problem has attracted progressive research attention over the years. There are broadly two categories of solutions to this problem: data-level solutions and algorithm-level solutions. The former either down-samples the majority class or up-samples the minority classes through various sampling strategies [4, 5, 6, 7]. The latter adjusts the training process or optimization objective [8, 9, 10]. Among the algorithm-level solutions, a highly recognized method is Focal Loss [10].

Focal Loss is a loss function that dynamically scales a Cross Entropy objective based on the classifier's uncertainty for each sample. The uncertainty is measured by the difference between the predicted probability and the ground truth label. Focal Loss naturally solves the class imbalance problem by emphasizing minority-class samples that have higher uncertainties during optimization. In addition to solving class imbalance problems, the Focal Loss philosophy – using uncertainty to re-weigh data samples – can be used to tackle more generic data disparity challenges [11, 12, 13, 14, 15, 16].

Despite all its merits, Focal Loss is not well-defined outside classification tasks because it uses the classifier output as an uncertainty measure which is not readily available in most non-classification tasks. One such task is keypoint detection, a popular and challenging field in computer vision. Keypoint detection has diverse applications, such as facial expression recognition [17, 18], human pose estimation [19, 20, 21], and medical diagnosis on anatomical abnormalities [22, 23]. Meanwhile, keypoint detection is known to suffer from class imbalance, scale imbalance, long-tailed data distributions, etc. [24, 25, 26]. Focal Loss would be a great solution to these problems, if only its utilities were not restricted to classification tasks.

To overcome this issue, we propose a simple yet effective solution named Adversarial Focal Loss (AFL) for keypoint detection. Rather than relying on existing model outputs, we leverage an external adversarial network which can holistically assess the difficulties of each data sample regardless of base model architecture. This in turn allows us to formulate a loss function that is semantically analogous to Focal Loss while cleanly extending to the keypoint detection problem. From our experiments, AFL can be seamlessly integrated into arbitrary loss functions. It minimally increases computational cost during training, and it adds no computational overhead during inference.

2 Related Work

Keypoint Detection The term "keypoint detection" has two slightly different meanings in computer vision. In the first definition, keypoint detection identifies *representative, yet not pre-defined* points in an image, usually over multiple frames of a video, for purposes such as registration or simultaneous localization and mapping (SLAM). This is usually done by using unsupervised image-processing techniques called keypoint detectors, with some of the most famous ones being SIFT [27], SURF [28], and ORB [29]. This definition is out of the scope of this paper.

In the second definition, which this paper focus on, keypoint detection identifies *pre-defined* points in an image that usually have precise meanings – for example, a set of recognizable points on a face or well-defined joints in a body. This is usually accomplished using supervised methods [30, 31, 32, 33, 34, 35, 36, 37, 38, 39].

Keypoint detection algorithms must answer two questions:

1. How to represent keypoint locations?
2. How to associate keypoints with objects when multiple objects are present?

In answer to the first question, while a few prior research directly performed regression on pixel coordinates [40, 41], the mainstream approach is to represent keypoints using feature maps (usually called "heatmaps") [30, 31, 32, 33, 34, 35, 36, 37, 38, 39]. In the feature-map approach, if we aim to predict a total of K target keypoints, we will predict K feature maps, one for each keypoint. Next, keypoint locations can be discovered through post-processing techniques, such as using the location of the highest-intensity pixel over each feature map. A keypoint is considered missing/non-existent if the corresponding feature map contains all zeros after thresholding.

In answer to the second question, keypoint detection approaches fall into two categories: top-down and bottom-up. Bottom-up approaches [34, 35, 36, 37] first detect all keypoints without caring about which object they belong to and then group the keypoints into objects. Top-down approaches [32, 33, 38, 39] identify objects first and then detect keypoints for each object. We demonstrate AFL on several common top-down approaches, though it could also be applied to bottom-up approaches.

Focal Loss Focal Loss, when it was first introduced [10], was presented as an upgrade to the Cross Entropy loss function in classification problems. Since the Cross Entropy loss comes with symmetric components, the authors of the Focal Loss paper used a simplification of notation as follows:

$$p_t = \begin{cases} p & \text{if ground truth label } y = 1 \\ 1 - p & \text{if ground truth label } y = 0 \end{cases} \tag{1}$$

where y is the one-hot encoded ground truth label and p is the predicted probability after the activation. It can be observed that $(1 - p_t)$ is the gap between the predicted probability score and the binary ground truth label, and hence it represents the uncertainty of the classifier on the input sample. With this notation, the Cross Entropy loss and the Focal Loss can be respectively written as CE (Eq. 2) and FL (Eq. 3).

$$\text{CE}(p_t) = -\log(p_t) \tag{2}$$

$$\begin{aligned} \text{FL}(p_t) &= -(1 - p_t)^\gamma \log(p_t) \\ &= (1 - p_t)^\gamma \, \text{CE}(p_t) \end{aligned} \tag{3}$$

The key element that sets Focal Loss apart from Cross Entropy is the introduction of the multiplicative coefficient $(1 - p_t)^\gamma$ which leverages the uncertainty $(1 - p_t)$. Easy examples lead to low uncertainty (i.e., when p is close to the ground truth label, or equivalently, when $(1 - p_t)$ is close to 0), whereas hard examples lead to high uncertainty. As a result, this coefficient selectively down-weighs the well-classified easy examples and up-weighs the high-uncertainty hard examples. γ is a tunable parameters that adjusts the strength of re-weighing.

Adversarial Networks The notion of an adversarial network was first conceptualized in the "Generative Adversarial Network" (GAN) paper [42]. A GAN consists of two components competing against each other, a generator $\boldsymbol{g}$ and a discriminator $\boldsymbol{d}$ (also known as the adversarial network). In its original formulation, the objective is to train $\boldsymbol{g}$ to generate realistic data, such as an image of a human face, from random noise. While a cohort of real data is available, there is no meaningful one-to-one mapping between the noise vectors and the real data. In absence of paired ground truth, the adversarial network $\boldsymbol{d}$ acts as the loss function. Iteratively, $\boldsymbol{d}$ is trained to distinguish between what $\boldsymbol{g}$ generated and the real data, and $\boldsymbol{g}$ is trained to generate data that look real enough to fool $\boldsymbol{d}$. If a delicate balance is well maintained over time, $\boldsymbol{g}$ can gradually generate data that are indistinguishable from the real distribution.

In this paper, different from the original purpose of GANs, we do not aim to perform data generation in any form. In our case, the generator $\boldsymbol{g}$ is replaced by the keypoint detection network $\boldsymbol{f}$, while the discriminator $\boldsymbol{d}$ is used to calculate the Adversarial Focal Loss. Specifically, we ask $\boldsymbol{d}$ to judge the quality of each predicted set of keypoints, which indirectly reflects the difficulty score of the corresponding sample. The difficulty score can then be used to create a loss function for keypoint detection that is semantically analogous to Focal Loss.

3 Adversarial Focal Loss

Adversarial Focal Loss (AFL) works in a similar way as the Focal Loss, except for one key difference: the way hard examples are discovered. Focal Loss uses the classification prediction as an indicator of uncertainty, whereas AFL leverages an independent discriminator to estimate the difficulty.

Vanilla Method for Keypoint Detection The workflow for vanilla keypoint detection is shown in Fig. 2 top. A main network $\boldsymbol{f}$ is trained to predict feature maps y' from the input image x. For each input x, a loss $L(x) = L(y, y')$ is computed between the prediction y' and the ground truth y. The loss is then used to update $\boldsymbol{f}$.

Adversarial Focal Loss The key of the proposed Adversarial Focal Loss method (see Fig. 2 bottom) is the discriminator $\boldsymbol{d}$. We use $\boldsymbol{d}$ to judge the quality of a feature representation. $\boldsymbol{d}$ is trained such that all model-generated representations are labeled as "low-quality" and all ground truth counterparts are labeled as "high-quality". Our formulation of the discriminator is inspired by Wasserstein GAN (WGAN) [43], a widely-used adversarial network formulation that improved the stability of learning compared to the traditional GAN [42].

Under the WGAN formulation [43], when a feature map y' is passed through $\boldsymbol{d}$, the resulting output is $\boldsymbol{d}(y') \in (-\infty, \infty)$. To generate a difficulty score $\in (0, 1)$, we squash $L_d^{y'} = -\boldsymbol{d}(y')$ by a sigmoid function (σ). The difficulty score $\sigma(L_d^{y'})$ is conceptually analogous to the uncertainty $(1 - p_t)$ in Focal Loss, so that it is multiplied with the regular loss L to form the Adversarial Focal Loss (Eq. 4). The intuition behind this formulation can be found in the subsequent section "Interpretation of the Loss".

$$\text{AFL}(x) = \text{stop\_gradient}(\sigma(L_d^{y'}))L(x) \tag{4}$$

As shown in Eq. 4, the gradient of the difficulty score is detached, which means the difficulty score is only used as a multiplicative coefficient in the AFL formulation.

Please note that when batch size > 1, the batch-averaging takes place over AFL rather than over $L_d^{y'}$, which ensures that each sample is weighed by its own difficulty score, not by the average difficulty score over that batch. The pseudo-code can be found in Appendix section A.1.

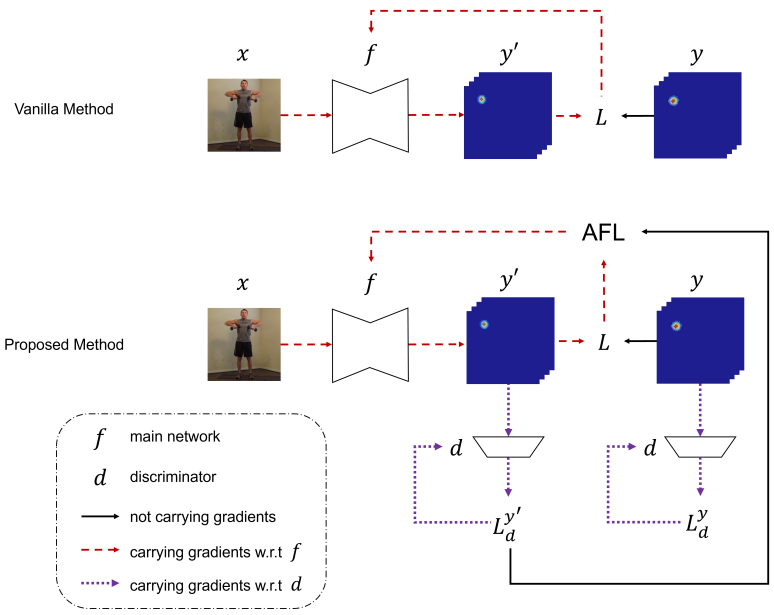

Figure 2: **Illustration of Adversarial Focal Loss (AFL).** AFL uses the discriminator ($\boldsymbol{d}$) to produce a difficulty score $\sigma(L_d^{y'})$ for each input sample. AFL subsequently utilizes the difficulty score to re-weigh the loss. This design is semantically analogous to Focal Loss even in absence of a classification objective. $L_d^{y'} = -\boldsymbol{d}(y') = -\boldsymbol{d}(\boldsymbol{f}(x))$; $L_d^{y} = -\boldsymbol{d}(y)$; $\sigma(\cdot)$: sigmoid function. In keypoint detection tasks only, feature representations y' and y are further condensed through a topology extractor (not shown in figure, see Appendix section A.2) before being given to $\boldsymbol{d}$.

Discriminator Since the specific architecture of the discriminator is not the main focus of this paper, we did not spend effort on carefully designing $\boldsymbol{d}$. Instead, we used a light-weight off-the-shelf discriminator from a publicly available implementation [44].

Following the WGAN training paradigm, we are using gradient penalty [45] to regularize the discriminator. The discriminator loss is defined as $L_d^y - L_d^{y'} + gp\_term$, where gp_term is the gradient penalty term [45]. The loss term $L_d^y - L_d^{y'}$ can be rewritten as $-\boldsymbol{d}(y) + \boldsymbol{d}(y')$. It guides the training in the same direction as a Cross Entropy objective $\text{CE}(p(y), 1) + \text{CE}(p(y'), 0)$ does, yet it offers a smoother gradient and higher stability during training [43].

Interpretation of the Loss Due to the training objective of $\boldsymbol{d}$, it learns to produce a larger $\boldsymbol{d}(y')$ if the representation appears of higher quality (in the sense that it looks like a ground truth representation) and a smaller $\boldsymbol{d}(y')$ otherwise. As a result, a low-quality representation (i.e., a hard example) would lead to a small $\boldsymbol{d}(y')$, thus a large $-\boldsymbol{d}(y')$, and eventually a difficulty score $\sigma(L_d^{y'})$ closer to 1. On the other hand, an easy example will yield a difficulty score closer to 0.

Topology Extractor We used a topology extractor to condense the geometric relations among keypoints within the same object, following the idea from [46]. It provides a more efficient representation of the keypoint patterns compared to the raw feature maps. Consequently, it reduces the computational cost. For details, see Appendix section A.2.

4 Empirical Studies[1]

Keypoint detection results on the MPII dataset We trained a widely-accepted baseline model [38] using the official implementation [47] with AFL integrated, on the MPII dataset [48]. The training settings are exactly the same as previously described [49]. The standard metric [48] is reported in Table 1. Compared to the baseline model, the adoption of AFL provides marginal yet consistent improvements on keypoint detection over all body parts.

Table 1: **Keypoint detection results evaluated on the MPII val dataset.**
Architecture acronyms: R-x = ResNet-x-FPN.

w/ AFL?	Arch	Head ↑	Shoulder ↑	Elbow ↑	Wrist ↑	Hip ↑	Knee ↑	Ankle ↑	Mean ↑
	R-50 [38]	96.4	95.3	89.0	83.2	88.4	84.0	**79.6**	88.5
✓	R-50 [38]	**96.9**	**95.5**	**89.1**	**83.7**	**88.9**	**84.5**	**79.6**	**88.9**

Keypoint detection results on the COCO dataset We trained 4 models of varying architectures and capacities [38, 39] using the official codebase [50] with AFL integrated on the COCO dataset [51]. The training settings are identical to what was previous described [39]. The standard metric, Average Precision scores, are reported in Table 2. This evaluation metric is based on Object Keypoint Similarity (OKS), a measure of keypoint detection accuracy. Details can be found in [39]. Specifically, AP^{50}/AP^{75} are the average precision at OKS=0.5/OKS=0.75, AP^M/AP^L are the average precision for medium/large objects, and AP is the mean average precision at 10 OKS positions evenly spaced between 0.5 and 0.95.

The adoption of AFL increased the mean AP score by 1.5 to 2.0 points across the 4 models. The subcategories AP^{50}, AP^{75}, and AP^M are also improved by varying amounts.

One intriguing observation is that despite the general improvements, AFL slightly suppresses the average precision on large objects (AP^L). We believe it's because the keypoints on large objects are generally easier to identify, and thus large objects are usually easy samples. As AFL emphasizes hard examples, performance on easy examples may be slightly sacrificed for an overall improvement.

Keypoint detection results on medical images Besides evaluations on standard keypoint detection benchmarks with natural images, we further examined the effectiveness of AFL on medical images.

[1] All experiments were completed on one NVIDIA A100 GPU, expect for two models (R-152 and H-48) for the COCO dataset which required two A100 GPUs.

Table 2: **Keypoint detection results evaluated on the COCO val dataset.** For rows with AFL, the additional computational costs during training are accounted for. We used a light-weight discriminator with 0.0097 M parameters and 0.00056 GFLOPs, and hence the reported Params and FLOPs are not affected. Improvements/degradations beyond 1.0 point are highlighted in green/red.
† Input image resized such that the short side is 800 pixels [32].
Architecture acronyms: R-x = ResNet-x-FPN; X-x=ResNeXt-x-FPN; H-x = HRNetV2p-Wx.

w/ AFL?	Arch	Input size	Params	FLOPs	AP ↑	AP^{50} ↑	AP^{75} ↑	AP^{M} ↑	AP^{L} ↑
	R-50 [32]	†800×800	-	-	65.1	86.6	70.9	59.9	73.6
	R-50 [52]	384×288	102 M	-	72.1	88.5	78.3	68.6	78.2
	R-101 [53]	224×224	51.8 M	-	66.5	87.3	72.8	-	-
	R-101 [32]	†800×800	-	-	66.1	87.7	71.7	60.5	75.0
	X-101 [32]	†800×800	-	-	70.4	89.3	76.8	65.8	78.1
	R-50 [38]	256×192	34.0 M	9.0 G	70.4	88.6	78.3	67.1	**77.2**
✓	R-50 [38]	256×192	34.0 M	9.0 G	**72.0** (+1.6)	**92.5** (+2.9)	**79.3** (+1.0)	**69.5** (+2.4)	76.1 (-0.9)
	H-32 [39]	256×192	28.5 M	7.1 G	74.4	90.5	81.9	70.8	**81.0**
✓	H-32 [39]	256×192	28.5 M	7.1 G	**76.1** (+1.7)	**93.6** (+3.1)	**83.5** (+1.6)	**73.2** (+2.4)	80.5 (-0.5)
	R-152 [38]	384×288	68.6 M	35.5 G	74.3	89.6	81.1	70.5	**81.6**
✓	R-152 [38]	384×288	68.6 M	35.5 G	**75.8** (+1.5)	**92.6** (+3.0)	**82.5** (+1.4)	**72.9** (+2.4)	80.4 (-1.2)
	H-48 [39]	384×288	63.6 M	32.9 G	76.3	90.8	82.9	72.3	**83.4**
✓	H-48 [39]	384×288	63.6 M	32.9 G	**78.3** (+2.0)	**93.6** (+2.8)	**84.9** (+2.0)	**75.4** (+3.1)	83.3 (-0.1)

We trained Feature Pyramid Networks [31] with and without AFL on a restricted X-ray dataset. The dataset defined 36 anatomical landmarks per patient, and was used to train keypoint detection models for diagnostic purposes (diagnostic task is not shown as it is outside the scope of this paper).

The major bottleneck of these keypoint detection models was the false negative rate. The version without AFL falsely generated a considerable number of low-response feature maps. With the adoption of AFL and everything else untouched, the number of false negatives was reduced by approximately a half (Table. 3).

Table 3: **Keypoint detection results evaluated on a medical dataset.** Models are evaluated on 51 X-ray images with keypoints annotated on 36 anatomical landmarks. These images are pre-processed through 4 different protocols leading to 4 different contrasts that are separately used for evaluation.

w/ AFL?	Arch	Contrast	Illustration	False Negative ↓	Total
	FPN [31]	higher		98	1827
✓	FPN [31]	higher		**27** (– 72%)	1827
	FPN [31]	high		85	1827
✓	FPN [31]	high		**32** (– 62%)	1827
	FPN [31]	low		89	1827
✓	FPN [31]	low		**43** (– 52%)	1827
	FPN [31]	lower		95	1827
✓	FPN [31]	lower		**69** (– 27%)	1827

Exploratory experiment: applying AFL for image classification As is evident by its design (Fig. 2), AFL has potentials beyond keypoint detection tasks. To demonstrate this potential, we evaluated AFL on image classification as a proof-of-concept.

We trained a wide-ResNet (WRN) [54] on CIFAR-100 [55] using Cross Entropy as the loss function, and keeping Focal Loss and AFL as variables for integration. The results are shown in Table 4.

For AFL in image classification, since the input to the discriminator was a one-dimensional probability vector instead of a matrix/image, we used a fully-connected discriminator instead of a convolutional network.

While we were unable to run extensive experiments on bigger datasets (e.g., ImageNet [56]) due to limited budget, our exploratory experiments demonstrate AFL's potential on tasks beyond keypoint detection. Focal Loss and AFL both provide noticeable benefits on top of Cross Entropy itself, yet the performance improvement from AFL is marginally bigger.

Interestingly, based on our experiments, using Focal Loss and AFL together does not lead to more favorable results. One possible reason could the over-emphasis of the harder examples while not focusing enough on easy examples.

Table 4: **Image classification results on CIFAR-100.** Each entry is averaged over 3 runs.

w/ AFL?	w/ Focal Loss?	Arch	Top-1 Accuracy ↑
		WSN-28-10, orig. [54]	79.50
		WSN-28-10, our impl.	82.14 $\pm$0.06
✓		WSN-28-10, our impl.	**82.86** $\pm$0.13
	✓	WSN-28-10, our impl.	82.38 $\pm$0.28
✓	✓	WSN-28-10, our impl.	81.83 $\pm$0.16

5 Behaviors of the difficulty score

As we claimed in section 3, we designed the difficulty score $\sigma(L_d^{y'})$ in our AFL formulation to serve as a multiplicative coefficient to up-weigh the hard examples and down-weigh the easy examples during training. We wanted to investigate whether it achieved this purpose. To that end, we examined the history of the difficulty score on easy versus hard examples during training.

Based on our speculation, we would anticipate the following trend:

1. At the very beginning of training, due to limited discriminative power, $\boldsymbol{d}$ will produce similar difficulty scores for all data samples. But the scores may cluster at any arbitrary value $\in (0, 1)$ depending on the initialization of $\boldsymbol{d}$.
2. As training progresses, $\boldsymbol{f}$ begins to generate higher-quality keypoint detections on easy examples before they can handle hard examples. At this stage, difficulty scores will start to diverge – the score will be lower for easy examples and higher for hard examples.

During the COCO training, we plotted the progression of the difficulty scores over time (Fig. 3). We randomly selected a few data samples to track their difficulty scores, and then grouped them into easy and hard examples based on the discriminator's response (Fig. 3, left and right). We find that $\boldsymbol{d}$ is able to judge the difficulty of data samples in a sensible manner. In general, easy examples (Fig. 3, left) exhibit well-exposed people in normal stances with keypoints visible over the full body. Hard examples (Fig. 3, right), on the other hand, are people that manifest unusual postures or have their body parts significantly occluded, which lead to abnormal keypoint relations or many missing keypoints.

The behaviors of the difficulty scores shown in our experiments are well aligned with our expectations, providing a glimpse into the underlying mechanism of the AFL method.

6 Conclusion and Future Works

In this paper, we first described the need for a Focal Loss equivalent in keypoint detection, and we explained why Focal Loss cannot be trivially adapted outside the classification domain. Then, we presented a novel approach called Adversarial Focal Loss (AFL) that is semantically equivalent to Focal Loss in non-classification tasks. We showed AFL's empirical effectiveness on three keypoint detection datasets and one image classification dataset. Finally, we investigated the behavior of AFL during training to verify its mechanism.

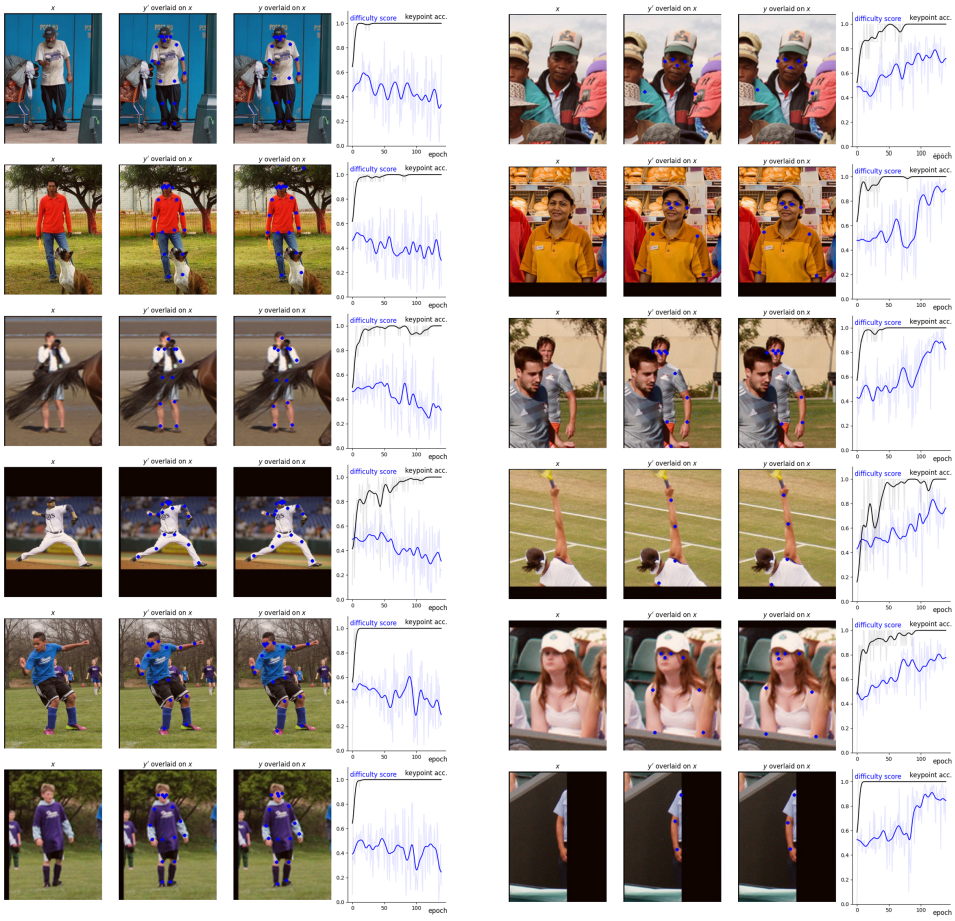

Figure 3: **Progression of the difficulty score $\sigma(L_d^{y'})$ for easy versus hard examples.** Left: easy examples; Right: hard examples. Within left/right, the 4 columns respectively represent input image, keypoint prediction, keypoint ground truth, and the difficulty score over epoch. Curves are Gaussian-smoothed for cleaner display, while the raw curves are shown in background with higher transparency. *It can be seen that a "hard example" does not necessarily imply bad performance.*

As for future work, we have identified two potential areas for further investigation.

Stronger regularization on the range of $L_d^{y'}$ The AFL method relies on the assignment of a proper difficulty score $\sigma(L_d^{y'})$ to each input sample. This consequently requires $L_d^{y'}$ to fall in a reasonable range. The current WGAN-like objective $L_d^y - L_d^{y'} + gp\_term$ does not pose a strict enforcement on this. For example, $\{L_d^y = -50, L_d^{y'} - 60\}$ and $\{L_d^y = 10, L_d^{y'} = 0\}$ both result in the same $L_d^y - L_d^{y'}$, but the former is more likely to produce difficulty scores close to 1 for all input samples, while the latter is more likely to produce diverse difficulty scores between 0 and 1. Admittedly, the gradient penalty term help regularize $L_d^y - L_d^{y'}$ towards zero-mean, but it is relatively weak and indirect. We believe a stronger regularization on the range of $L_d^{y'}$ may be preferable.

Extending AFL to other tasks It is easy to notice that AFL can be applied to a wider domain than Focal Loss thanks to its design (Fig. 2). Besides tasks that we have already studied, any task that follows the formulation in the upper section of Fig. 2 may find AFL beneficial. This would include denoising, semantic segmentation, superresolution, domain adaptation, among many others. A particularly interesting possibility is AFL in GANs, i.e., using AFL to dynamically adjust per-sample generator loss in tasks such as image generation, style transfer, etc. While this may lead to complications in optimization, the formulation of AFL shows no objection to such challenges.

References

[1] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 3485–3492. IEEE, 2010.

[2] Brian Mac Namee, Padraig Cunningham, Stephen Byrne, and Owen I Corrigan. The problem of bias in training data in regression problems in medical decision support. *Artificial intelligence in medicine*, 24(1):51–70, 2002.

[3] Claire Cardie and Nicholas Howe. Improving minority class prediction using case-specific feature weights. In *International Conference on Machine Learning*, 1997.

[4] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259, 2018.

[5] Jason Van Hulse, Taghi M Khoshgoftaar, and Amri Napolitano. Experimental perspectives on learning from imbalanced data. In *Proceedings of the 24th international conference on Machine learning*, pages 935–942, 2007.

[6] Inderjeet Mani and I Zhang. knn approach to unbalanced data distributions: a case study involving information extraction. In *Proceedings of workshop on learning from imbalanced datasets*, volume 126, pages 1–7. ICML, 2003.

[7] Miroslav Kubat, Stan Matwin, et al. Addressing the curse of imbalanced training sets: one-sided selection. In *Icml*, volume 97, page 179. Citeseer, 1997.

[8] Bartosz Krawczyk. Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, 5(4):221–232, 2016.

[9] Charles X Ling and Victor S Sheng. Cost-sensitive learning and the class imbalance problem. *Encyclopedia of machine learning*, 2011:231–235, 2008.

[10] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[11] Giang Son Tran, Thi Phuong Nghiem, Van Thi Nguyen, Chi Mai Luong, and Jean-Christophe Burie. Improving accuracy of lung nodule classification using deep learning with focal loss. *Journal of healthcare engineering*, 2019, 2019.

[12] Jishnu Mukhoti, Viveka Kulharia, Amartya Sanyal, Stuart Golodetz, Philip Torr, and Puneet Dokania. Calibrating deep neural networks using focal loss. *Advances in Neural Information Processing Systems*, 33:15288–15299, 2020.

[13] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European conference on computer vision (ECCV)*, pages 734–750, 2018.

[14] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9627–9636, 2019.

[15] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. *Advances in neural information processing systems*, 31, 2018.

[16] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019.

[17] Xiangxin Zhu and Deva Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *2012 IEEE conference on computer vision and pattern recognition*, pages 2879–2886. IEEE, 2012.

[18] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep convolutional network cascade for facial point detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3476–3483, 2013.

[19] Zigang Geng, Ke Sun, Bin Xiao, Zhaoxiang Zhang, and Jingdong Wang. Bottom-up human pose estimation via disentangled keypoint regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14676–14686, 2021.

[20] Adrian Bulat and Georgios Tzimiropoulos. Human pose estimation via convolutional part heatmap regression. In *European Conference on Computer Vision*, pages 717–732. Springer, 2016.

[21] Xiao Chu, Wei Yang, Wanli Ouyang, Cheng Ma, Alan L Yuille, and Xiaogang Wang. Multi-context attention for human pose estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1831–1840, 2017.

[22] Jun Zhang, Mingxia Liu, and Dinggang Shen. Detecting anatomical landmarks from limited medical imaging data using two-stage task-oriented deep neural networks. *IEEE Transactions on Image Processing*, 26(10):4753–4764, 2017.

[23] Qingsong Yao, Quan Quan, Li Xiao, and S Kevin Zhou. One-shot medical landmark detection. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 177–188. Springer, 2021.

[24] Kemal Oksuz, Baris Can Cam, Sinan Kalkan, and Emre Akbas. Imbalance problems in object detection: A review. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3388–3415, 2020.

[25] Yu Li, Tao Wang, Bingyi Kang, Sheng Tang, Chunfeng Wang, Jintao Li, and Jiashi Feng. Overcoming classifier imbalance for long-tail object detection with balanced group softmax. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10991–11000, 2020.

[26] Yuan Yao, Yasamin Jafarian, and Hyun Soo Park. Monet: Multiview semi-supervised keypoint detection via epipolar divergence. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 753–762, 2019.

[27] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[28] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[29] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. Ieee, 2011.

[30] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European conference on computer vision*, pages 483–499. Springer, 2016.

[31] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[32] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[33] Yilun Chen, Zhicheng Wang, Yuxiang Peng, Zhiqiang Zhang, Gang Yu, and Jian Sun. Cascaded pyramid network for multi-person pose estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7103–7112, 2018.

[34] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7291–7299, 2017.

[35] Alejandro Newell, Zhiao Huang, and Jia Deng. Associative embedding: End-to-end learning for joint detection and grouping. *Advances in neural information processing systems*, 30, 2017.

[36] George Papandreou, Tyler Zhu, Liang-Chieh Chen, Spyros Gidaris, Jonathan Tompson, and Kevin Murphy. Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model. In *Proceedings of the European conference on computer vision (ECCV)*, pages 269–286, 2018.

[37] Muhammed Kocabas, Salih Karagoz, and Emre Akbas. Multiposenet: Fast multi-person pose estimation using pose residual network. In *Proceedings of the European conference on computer vision (ECCV)*, pages 417–433, 2018.

[38] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *European Conference on Computer Vision (ECCV)*, 2018.

[39] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3349–3364, 2020.

[40] Aiden Nibali, Zhen He, Stuart Morgan, and Luke Prendergast. Numerical coordinate regression with convolutional neural networks. *arXiv preprint arXiv:1801.07372*, 2018.

[41] Zhengxiong Luo, Zhicheng Wang, Yan Huang, Liang Wang, Tieniu Tan, and Erjin Zhou. Rethinking the heatmap regression for bottom-up human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13264–13273, 2021.

[42] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[43] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[44] E. Linder-Norén. Pytorch implementations of generative adversarial networks. `https://github.com/eriklindernoren/PyTorch-GAN`, 2018.

[45] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

[46] Shaobo Zhang, Wanqing Zhao, Ziyu Guan, Xianlin Peng, and Jinye Peng. Keypoint-graph-driven learning framework for object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1065–1073, 2021.

[47] X. Bin. Simple baselines for human pose estimation and tracking. `https://github.com/Microsoft/human-pose-estimation.pytorch`, 2018.

[48] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition*, pages 3686–3693, 2014.

[49] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *Proceedings of the European conference on computer vision (ECCV)*, pages 466–481, 2018.

[50] X. Bin. Deep high-resolution representation learning for human pose estimation (cvpr 2019). `https://github.com/leoxiaobin/deep-high-resolution-net.pytorch`, 2019.

[51] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[52] Gyeongsik Moon, Ju Yong Chang, and Kyoung Mu Lee. Posefix: Model-agnostic general human pose refinement network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7773–7781, 2019.

[53] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018.

[54] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[55] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *technical report* *https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf*, 2009.

[56] 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. Ieee, 2009.

A Appendix

A.1 Pseudo-code of the vanilla method for keypoint detection versus our proposed AFL

Algorithm 1: Vanilla Method

Required :
input images X
ground truth feature maps Y
main model $\boldsymbol{f}$

for $(x, y) \in (X, Y)$ **do**
  $y' \leftarrow \boldsymbol{f}(x)$
  $L \leftarrow$ loss_function(y, y')
  $\boldsymbol{f}$ updates w.r.t. L
end

Algorithm 2: Proposed AFL Method

Required :
input images X
ground truth feature maps Y
main model $\boldsymbol{f}$
discriminator $\boldsymbol{d}$
Optional :
topology extractor $\boldsymbol{t}$ (required for keypoint detection)

for $(x, y) \in (X, Y)$ **do**
  $y' \leftarrow \boldsymbol{f}(x)$
  $L \leftarrow$ loss_function(y, y')

  $/ *$ The following 2 lines are only relevant to keypoint detection. Skip them otherwise. $* /$
  $y \leftarrow \boldsymbol{t}(y)$
  $y' \leftarrow \boldsymbol{t}(y')$

  $(L_d^y, L_d^{y'}, \text{gp\_term}) \leftarrow$ WGAN_GP$(\boldsymbol{d}, y, y')$
  $L_d \leftarrow L_d^y - L_d^{y'} + \text{gp\_term}$
  AFL $\leftarrow$ stop_gradient$(\sigma(L_d^{y'})) \cdot L$

  $\boldsymbol{d}$ updates w.r.t. L_d
  $\boldsymbol{f}$ updates w.r.t. AFL
end

$/ *$ Detail of WGAN_GP [45] (Using the default setting: $\lambda = 10$) $* /$

procedure WGAN_GP($\boldsymbol{d}, y, y'$)
  $\alpha \sim U[0, 1]$
  $y_{\text{mixup}} \leftarrow \alpha \cdot y + (1 - \alpha) \cdot y'$
  $L_d^y \leftarrow -\boldsymbol{d}(y)$
  $L_d^{y'} \leftarrow -\boldsymbol{d}(y')$
  gp_term $\leftarrow \lambda \cdot (||\nabla_{y_{\text{mixup}}} \boldsymbol{d}(y_{\text{mixup}})||_2 - 1)^2$
return $L_d^y, L_d^{y'}$, gp_term

A.2 Topology extractor

The topology extractor is a mapping from the feature space $\mathbb{R}^{W\times H\times K}$ (for K keypoints) to a set of N adjacency matrices $\mathbb{R}^{K\times K\times N}$. In our current formulation we have $N = 2$, where the first adjacency matrix represents the planar affinity of any two keypoints and the second matrix represents their angular affinity.

In practice, we first instantiate two matrices M_p and M_a, both of dimensionality $\mathbb{R}^{K\times K}$, to respectively represent the planar and angular affinities. We then find the coordinates of the centroids of these keypoints $(x, y)_i$ for $i \in \mathbb{K}$, where $\mathbb{K} := [1, 2, ..., K]$. Any keypoint j that does not exist on the feature map and thus does not have a centroid will be ignored, such that $M(j, i)$ and $M(i, j)$ will remain 0 for all $i \in \mathbb{K}$. For notational simplicity, we define $\mathbb{K}^+$ as the subset of $\mathbb{K}$ where keypoints exist.

For planar affinity, we define

$$M_p(i,j) = 1 - \frac{||(x,y)_i - (x,y)_j||_2}{||(W,H) - (0,0)||_2} \tag{5}$$

For angular affinity, we define

$$M_a(i,j) = \frac{1}{2} + \frac{1}{2}\cos\angle A(i,j) \tag{6}$$

To find $\angle A(i, j)$, we first compute the coordinates of the "global centroid" among all existing keypoints, $(x, y) = \left(\frac{1}{|\mathbb{K}^+|}\sum_{k\in\mathbb{K}^+} x_k, \frac{1}{|\mathbb{K}^+|}\sum_{k\in\mathbb{K}^+} y_k\right)$. Then, we define $\angle A(i, j)$ for each $i, j \in \mathbb{K}^+$ as an angle whose vertex is the "global centroid" (x, y) and whose two sides pass through $(x, y)_i$ and $(x, y)_j$, respectively.

A.3 Additional cases for section 5

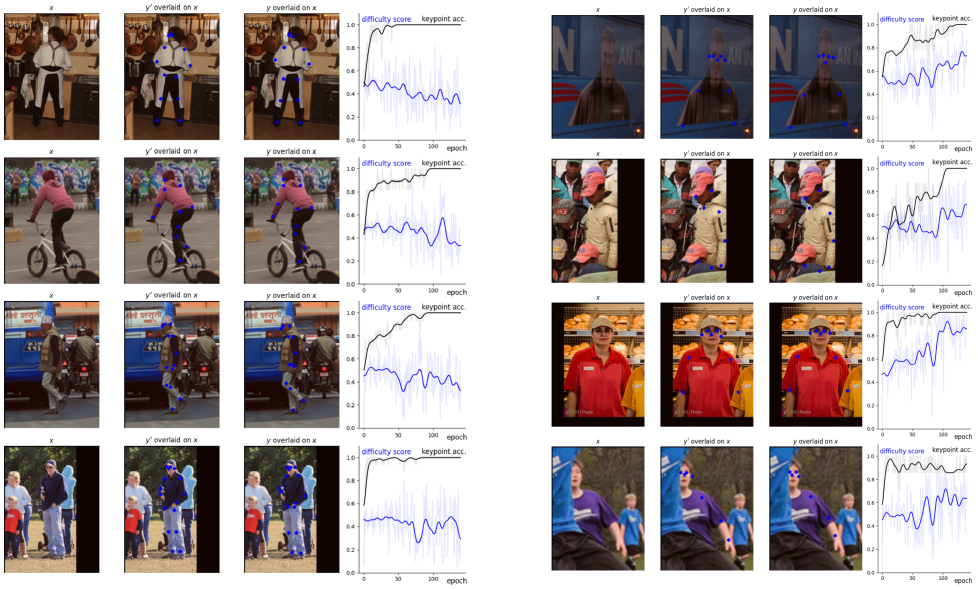

Figure 4: **Progression of the difficulty score $\sigma(L_d^{y'})$ for easy versus hard examples: additional cases.** Left: easy examples; Right: hard examples. Within left/right, the 4 columns respectively represent input image, keypoint prediction, keypoint ground truth, and the difficulty score over epoch. Curves are Gaussian-smoothed for cleaner display, while the raw curves are shown in background with higher transparency. *Again, it can be seen that a "hard example" does not necessarily imply bad performance.*

A.4 Failure cases for section 5

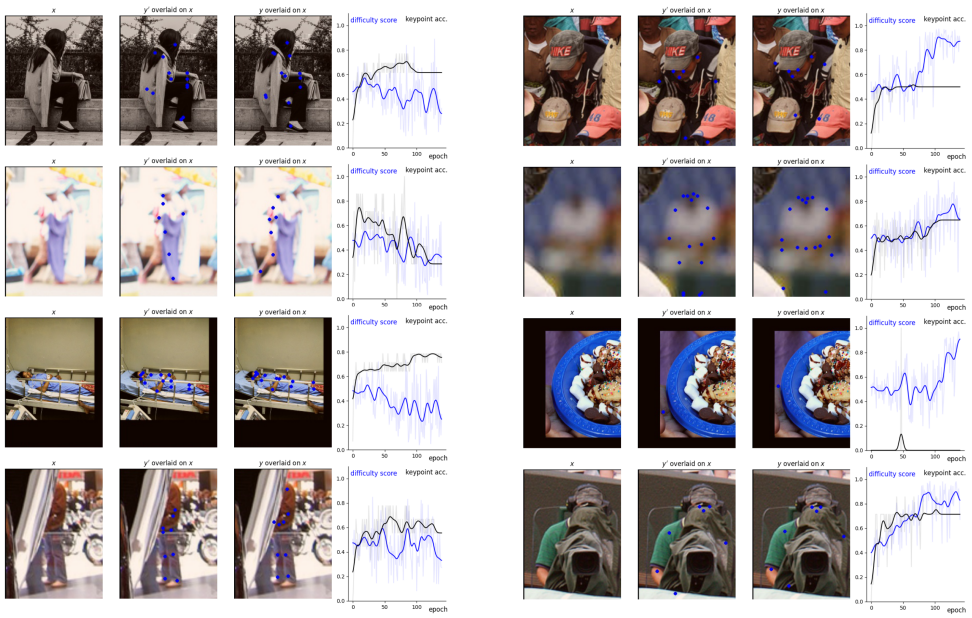

Figure 5: **Progression of the difficulty score $\sigma(L_d^{y'})$ for easy versus hard examples: failure cases. Examples with low keypoint detection accuracy are selected for this figure.** Left: easy examples; Right: hard examples. Within left/right, the 4 columns respectively represent input image, keypoint prediction, keypoint ground truth, and the difficulty score over epoch. Curves are Gaussian-smoothed for cleaner display, while the raw curves are shown in background with higher transparency. Just like a "hard example" does not necessarily imply bad performance, *an "easy example" does not necessarily imply good performance.*