Mixture Dense Regression for Object Detection and Human Pose Estimation

Ali Varamesh, Tinne Tuytelaars
ESAT-PSI, KU Leuven
{ali.varamesh,tinne.tuytelaars}@esat.kuleuven.be

Abstract

Mixture models are well-established learning approaches that, in computer vision, have mostly been applied to inverse or ill-defined problems. However, they are general-purpose divide-and-conquer techniques, splitting the input space into relatively homogeneous subsets in a data-driven manner. Not only ill-defined but also well-defined complex problems should benefit from them. To this end, we devise a framework for spatial regression using mixture density networks. We realize the framework for object detection and human pose estimation. For both tasks, a mixture model yields higher accuracy and divides the input space into interpretable modes. For object detection, mixture components focus on object scale, with the distribution of components closely following that of ground truth the object scale. This practically alleviates the need for multiscale testing, providing a superior speed-accuracy tradeoff. For human pose estimation, a mixture model divides the data based on viewpoint and uncertainty – namely, front and back views, with back view imposing higher uncertainty. We conduct experiments on the MS COCO dataset and do not face any mode collapse.

1. Introduction

Over the span of a few years, there has been massive progress in designing increasingly efficient architectures, loss functions, and optimization procedures for mainstream computer vision tasks such as image classification, object detection, semantic segmentation, and pose estimation [18, 37, 39, 31, 25, 16, 3, 14, 35]. However, from a machine learning perspective, there is still a lot to be desired. For example, when it comes to capturing the multi-modal nature of visual data, most of the solutions for object detection leave it to the optimizer to figure it all out. Still, given the fundamental limitations of machine learning [41, 40], this is an unrealistic expectation. Modeling a multi-modal distribution using a single-mode model will always lead to sub-optimal predictions.

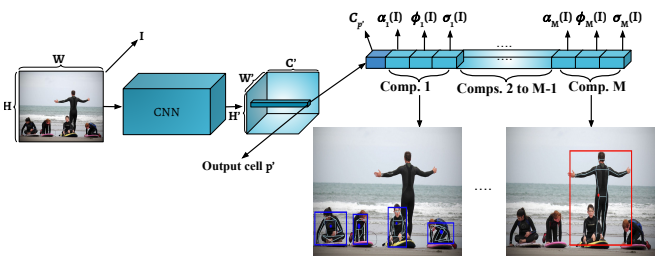As a case in point, let us consider dense object detection.

Figure 1: The proposed mixture spatial regression framework. When realized for human pose estimation, two modes are retrieved based on viewpoint.

For a given input image, at each spatial location, a model should have a classification output and do a spatial regression. The classification component naturally is a multi-modal problem. As such, any solution has to learn different modalities, commonly realized via multinomial classification. For the spatial regression, on the other hand, either there are no such discernible modalities, or it is not straightforward how to model them. In object detection, it could be the categories that govern the bounding box regression, mere foreground vs. background segmentation, or a coarser categorization. We cannot say for sure which one it is. Similarly, in dense human pose estimation using offset regression, there is a separate output for each body part. But, given a part, either the scale or the pose of a person could be the dominant mode for the regression task.

As one can see, explicitly identifying the underlying modes is often impossible. However, in machine learning, there are well-established techniques for dealing with multi-modality. For example, mixture models [29], including mixture density networks [4] and mixture of experts [15], are powerful techniques for imposing structure on a model's prediction mechanism, which leads to higher accuracy and model interpretability. In particular, they divide the input space based on the relationship between input and output. In other words, depending on the targets, they optimally split the input space so that higher performance can be achieved. Of course, these techniques are not unknown to our community, but so far, they have been almost exclu-

sively applied to ill-defined problems like 3D pose estimation [22, 42] or video frame prediction [28].

In this work, we advocate a more extensive use of mixture models. We show how to improve bottom-up dense object detection and human pose estimation tasks by incorporating their spatial regression sub-tasks in a mixture density network. Our framework yields significant improvements for both tasks in terms of accuracy, speed-accuracy trade-off, convergence, and interpretability. To the best of our knowledge, we are the first to successfully integrate mixture density networks into 2D object detection and human pose estimation tasks. We have published the source-code[1]. The following is a summary of our contributions:

- To account for the multi-modal nature of the visual domain, we propose a new formulation for dense spatial localization using mixture density networks, which proves to be superior to single-mode models.

- We show that a mixture object detection model learns to deal with object scale variation through different components, provides significantly better speed-accuracy trade-off, and converges faster.

- We model offset regression in human pose estimation using a mixture model. Our model yields significant gain in accuracy and reveals viewpoint as the dominant multi-modal factor. Further analysis shows that, in fact, uncertainty decides mixture components.

2. Related work

Modern solutions for object detection and human pose estimation either use a top-down design or a bottom-up one. Here, we review related work from both approaches. However, our framework is based on the bottom-up approach.

2.1. Top-down models

In top-down models, an image is first processed by a CNN to propose an initial set of regions that may include objects of interest. The proposals are then further processed for more accurate recognition and localization. For object detection, this means classification of the object in a given region and generating a tight bounding box around it [10, 35, 13, 38]. Before the second stage, the regions are resized to a fixed size, therefore gaining some built-in robustness to scale variation. In top-down human pose estimation, first, a set of regions containing persons are generated by an object detector; next within each region, a predefined set of body keypoints (e.g. eyes, shoulders, etc.) are localized [31, 23, 13, 6, 23]. In general, two-stage procedures are more accurate than single-stage models but incur a significant delay. A state-of-the-art model can may about a second to process an image[24].

[1]https://github.com/alivaramesh/MixtureDenseRegression

2.2. Bottom-up models

In a bottom-up approach, in a single stage, a model simultaneously does classification at all given spatial locations in a dense fashion and also estimates pose parameters. Classification head determines if a location indicates the center of an object, or if a location is inside an object region. In object detection, desired pose parameters represent object bounding boxes [27, 34, 25, 44].

In bottom-up human pose estimation, the traditional approach is to generate a dense heatmap for each body part to predict the presence of that part at each spatial location. Simultaneously, at each location, an embedding is generated to distinguish keypoints of different person instances [5, 32, 30]. In the other approach, every location is classified as center of a person or not, and an offset vector is generated from that location to each body part [44]. This method is faster and eliminates the sub-optimal post-processing step for grouping keypoints using embeddings generated by the former approach. However, it is more challenging to optimize. In fact, in [44], offset regression does not deliver high spatial precision, and body part heatmaps are used to refine the predictions, incurring a delay. A central motivation for our work has been to improve offset regression such that a refinement step becomes unnecessary.

2.3. Multiple choice models

Multiple choice models include approaches where, for a given input, a model makes multiple predictions, from which one is chosen as the best. In spirit, they are similar to mixture models. In the context of image classification, it is shown by many works that generating multiple diverse predictions works better than a single head or an ensemble of models [12, 21, 20, 36]. However, they depend on an oracle to choose the best prediction for a given input. This may be fine when another downstream application further examines the predictions, but it is a big obstacle for the widespread deployment of such models. Additionally, unlike mixture density networks, these methods do not have a mechanism to learn the density of outputs conditioned on the input.

Mixture density networks [4] have attracted a lot of attention in recent years. In particular, it has been applied to 3D human pose estimation [22] and 3D hand pose estimation [42]. In 2D human pose estimation, [36] have reported an unsuccessful application of MDNs, failed due to numerical instabilities. Here, we show that properly modifying the variance activation function for Gaussian kernels eliminates such instabilities. MDNs are also used by Prokudin et al. [33] for quantifying uncertainty in angular pose estimation. Nevertheless, to the best of our knowledge, mixture density networks have not been adapted for mainstream vision tasks of object detection and human pose estimation on large scale real-world datasets.

3. Methodology

In this section, first, we review the mixture density networks. Next, we illustrate how to model dense spatial regression using a mixture model.

3.1. Mixture Density Networks

Mixture models are powerful tools for estimating the density of any distribution [29]. Ideally, they recover modes that contribute to the generation of data and their distribution. For a regression task, mixture models help to avoid converging to an average target given an input. For example, consider estimating the density of data generated by a bi-modal Gaussian distribution. Using a single Gaussian model will deliver sub-optimal results by predicting a mean value squashed in between two actual centers. However, a mixture model avoids this issue by assigning data points to proper generators. As in the example above, mixture models are straightforward to interpret.

In the context of neural networks, mixture density networks (MDN) [4] enable us to use a neural network to estimate the parameters of a mixture model. An MDN estimates the probability density of a target vector conditioned on the input. Assume a regression task on a dataset with the set of input vectors denoted by $\{\mathbf{x}_0 \ldots \mathbf{x}_n\}$ and the associated target vectors $\{\mathbf{t}_0 \ldots \mathbf{t}_n\}$. The objective of MDN is to fit the weights of a neural network such that it maximizes the likelihood of the training data. The key issue then is to formulate the probability density of the target conditioned on the input. Eq. 1 shows how this is done in MDNs.

$$p(\mathbf{t}_i|\mathbf{x}_i) = \sum_{m=1}^{M} \alpha_m(\mathbf{x}_i)\phi_m(\mathbf{t}_i|\mathbf{x}_i) \qquad (1)$$

In Eq. 1, M is a hyper-parameter denoting the number of components constituting the mixture model. $\alpha_m(\mathbf{x}_i)$ is called mixing coefficient and indicates the probability of component m being responsible for generation of the sample $\mathbf{x}_i$. ϕ_m is the probability density function of component m for computing density of $\mathbf{t}_i$ conditioned on $\mathbf{x}_i$. The density function can be chosen from a wide set of well-known kernels. In practice Gaussian kernel (given in Eq. 2) works well and is the most common choice.

$$\phi_m(\mathbf{t}_i|\mathbf{x}_i) = \frac{1}{(2\pi)^{c/2}\sigma_m(\mathbf{x}_i)^c}exp\Big\{ - \frac{||\mathbf{t}_i - \boldsymbol{\mu}_m(\mathbf{x}_i)||^2}{2\sigma_m(\mathbf{x}_i)^2}\Big\} \qquad (2)$$

In Eq. 2, c indicates dimension of the target vector, $\boldsymbol{\mu}_m$ is the mean of component m, and σ_m is the common variance parameter for component m. The variance term does not have to be shared between dimensions of target space and can be replaced with a diagonal or full covariance matrix if necessary [4]. Note that MDNs do not presume independence among the components of the target vector $\mathbf{t}$. To be more precise, elements of the $\mathbf{t}$ are independent given a mixture component; however, the full model enforces dependence among the elements of the target, through learning different modes of the data using each mixture component.

3.2. Mixture dense spatial regression

In this section, we illustrate how to formulate both object detection and human pose estimation tasks using mixture density networks. We develop our formulation on top of the recent CenterNet dense detection model [44]. The general formulation is as follows. Given an image, each spatial location needs to be classified to determine whether it is representing the center of an instance. The classification branch is realized by doing dense binary classification for each category $y \in \mathbf{Y}$. The number of categories for object detection is equal to the number of classes in a dataset, and for human pose estimation, it only includes the person class. Beside the classification branch, at each location we also need to predict pose parameters of interest $\mathbf{T}$ [1]. For object detection, pose parameters correspond to the height and width of an object, therefore a 2-dimensional vector. For human pose estimation, $\mathbf{T}$ includes K $2D$ dimensional offset vectors from the center of a person to each of the K body parts (K is 17 in the MS COCO keypoints dataset). This formulation is, in particular, efficient for human pose estimation. Unlike the top-down methods, there is no need to use an object detector to localize person instances first. And, unlike traditional bottom-up methods, the grouping of body parts is not left as a post-processing step based on learned embeddings. Rather, at every spatial location, the model predicts if it is the center of a person and generates an offset vector to the location of each keypoint.

The most common loss for training the spatial pose parameters is the L_1 loss function [32, 17, 5, 44]. However, spatial regression is a multi-modal task, and we believe modeling it using a single-mode network will lead to a sub-optimal prediction. Therefore, we use a mixture density network to model the spatial regression task. Now we proceed to describe our mixture dense prediction model formally.

Given an input RGB image I of size $H * W * 3$, a CNN processes I and generates an output of dimensions $H' * W' * C'$. Here we have $H = D * H'$ (and similarly for width), where D is the down sampling factor of the network. We indicate the set of all output cells using P. At $p \in P$, the output channels C', includes Y binary classification channels. It also includes pose parameters T. For object detection, T is a 2D vector of the form: $T = [p_w, p_h]$ corresponding to width and height of an object. For human pose estimation, T includes K 2D offset vectors from the person center to each keypoint, that is $T = [o_{p,x}^0, o_{p,y}^0, \ldots, o_{p,x}^{K-1}, o_{p,y}^{K-1}]$. The ground truth pose parameters are denoted using $\hat{T}$. Once the network classifies p as centre of an instance, the pose parameters are

readily available to generate the complete prediction.

We adapt Eq. 1 and Eq. 2 such that the mixture model predicts pose parameters T. That is, if we have an M component mixture model (MDN_M), $\boldsymbol{\mu}_m$ would represent pose parameters predicted by component $m \in M$. Then the density of the ground truth pose parameters $\hat{T}$ conditioned on image I is given by Eq. 3, where the density function ϕ_m for each mixture component is given by Eq. 4. In Eq. 4, $T_m(I)$ is the input dependent pose parameters generated by component m. $\sigma_m(I)$ is the standard deviation of the component m in two dimensions, that is, X and Y for horizontal and vertical axes. To account for keypoints' scale difference, in our implementation of Eq. 4 for human pose estimation, for each keypoint we divide $\sigma_m(I)$ by its scale factor provided in the COCO dataset.

$$p(\hat{T}|I) = \sum_{m=1}^{M} \alpha_m(I)\phi_m(\hat{T}|I) \qquad (3)$$

$$\phi_m(\hat{T}|I) = \frac{1}{(2\pi)^{c/2}\sigma_m(I)^c} exp\left\{-\frac{||\hat{T} - T_m(I)||^2}{2\sigma_m(I)^2}\right\} \quad (4)$$

Given the conditional probability density of the ground truth in Eq. 3, we can define the regression objective as the negative log-likelihood and minimize it using stochastic gradient descent. In Eq. 5, we provide the negative log-likelihood for the pose targets generated by MDN, where N is the number of samples in the dataset. Essentially, this loss term replaces the popular L_1 loss for pose regression targets. Note that we implement MDN in a dense fashion. That is, density estimation is done independently at each spatial location $p' \in P'$. A schematic overview of the model is shown in Fig. 1.

$$L_T = \sum_{i=1}^{N} -\ln \sum_{m=1}^{M} \alpha_m(I_i)\phi_m(\hat{T}_i|I_i) \qquad (5)$$

We do not modify the other loss terms used in Center-Net. This includes a binary classification loss L_C for each class, offset regression loss term $L_{C_{off}}$ to compensate for lost spatial precision due to downsampling, and the term L_T loss for pose parameters. The total loss is given in Eq. 6:

$$L_{total} = \lambda_C L_C + \lambda_{off} L_{C_{off}} + \lambda_T L_T \qquad (6)$$

In the case of human pose estimation, CenterNet also adds a traditional heatmap based keypoint detection and small offset regression heads to the network. This is used for further refinement at inference. These loss terms will be denoted by L_{HM} and $L_{KP_{off}}$, respectively.

In our experiments, we have $\lambda_T = 0.1$. It is tuned such that the performance of MDN_1 is on par with that of CenterNet (which is a single-mode model). Other loss weights are the same as the ones used in CenterNet, that is $\lambda_C = 1$, $\lambda_{off} = 0.1$, and $\lambda_{HM} = 1$.

3.3. Inference

Once the network is trained, at each spatial location, the classification branch determines if it is the center of an instance (we use the bounding box center for ground truth). If yes, we can use either the mixture of outputs by the components or the one with the highest score to generate the pose parameters. We tried both cases and found out that using the maximum component leads to slightly better results.

4. Experiments

We have conducted extensive experiments on the MS COCO 2017 dataset [26]. From the training split (*coco-train*), we use all the 118k images for object detection and the images with at least one person instance (64k images) for human pose estimation. To compare against the state-of-the-art, we use the COCO test-dev split (*coco-test-dev*). All other evaluations are done on COCO validation split which contains 5k images (*coco-val*). The evaluation metric in all experiments is COCO average precision (AP). For object detection we experiment using hourglass-104 (HG) [19] and deep layer aggregation (DLA34) [43] architectures. For human-pose estimation, we only use HG.

4.1. Training

We use the ADAM optimizer [16] for training our models. For models based on HG, we use batch size 12 and three different schedules taking 50 (1X), 100(2X), and 150(3X) epochs. We initialize the learning rate to 2.5e-10 and drop it by factor 10 at the tenth epoch from the last. For the ones based on DLA34, we train for 140 epochs (1X) with batch size 32 and learning rate 2e-4, which is dropped by factor 10 at epochs 90 and 120. These settings are analogous to the ones used in CenterNet [44]. Unless indicated otherwise, for all experiments, we use HG with the 1X schedule. To allow for proper comparisons, we train all models, including the single-mode base model, CenterNet, from scratch.

4.2. Activation function for variance terms

A typical formulation for Gaussian MDNs uses an activation function for variance terms that generates positive values [9, 36, 22]. However, this causes numerical instabilities when some of the mixture components are redundant or do not have a significant contribution. Unlike the significant modes, the variance term of these components do not get trained to be in a suitable range, and may lie between zero and one. If only a small portion of samples get assigned to them at training, highly irregular gradients will be generated, impeding proper training of the model. For example, average precision on *coco-val* can fluctuate between 5 and 30 percent in consecutive epochs even after being trained for tens of epochs. A simple remedy is to prevent the variance term from falling in the range $(0, 1)$. Hence, we
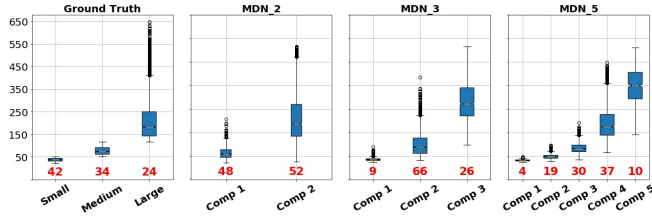
Figure 2: Distribution of ground truth (GT) object scales compared to the distribution of scale by mixture models trained on HG architecture. Components are named in order of their scale range. Below the boxes, we also indicate the distribution of GT instances and mixture components.

use a modified version of exponential linear units (ELU) [8] for activation of the variance terms such that the minimum value is one. We experimented with larger minimum values (up to 10) but did not observe any significant difference.

4.3. Object Detection

In table 1 we provide evaluations on the *coco-val* set for the baseline and mixture models with two to five components on HG architecture. With the 1X schedule, MDN_3 achieves an impressive 3.1 percentage points improvement.

4.3.1 Analysis of the components

To gain insight into what each component learns, we visually investigated the objects regressed by each component (samples can be seen in Fig. 5a). It turns out that MDN separates the dataset based on the objects' scale. To confirm this in quantitative terms, we looked at the distribution of scale for objects regressed by different components. When we compare the scale distribution of the ground truth data, we observe a strong correlation, as shown in Fig. 2. Quantitatively, the Pearson correlation coefficients for maximum component and the scale on *coco-val* is 0.76. The correlation between components and categories is only 0.04. In other words, there is no obvious relationship between components and categories.

As a further test, we trained a version of the base model with separate box prediction heads for each category. However, we did not observe any notable difference compared to the case where there is only a shared box prediction head. Therefore, merely using category-specific prediction heads does not have any benefit.

Based on table 1, the number of components does not seem to have a significant effect when increased to more than three. However, according to Fig. 2, it does lead to better separation of the data. The fact that MDN_5 yields better separation, but not higher accuracy, seems odd. We believe this could be because the classification branch is not keeping up with the regression head. This is an interesting

Model	Comp.	AP	AP'50	AP'75	AP'S	AP'M	AP'L
Single mode base	-	35.9	54.5	38.4	20.4	39.5	46.1
MDN_2	All	38.4	56.6	41.2	22.3	42.4	49.9
	1	27.8	42.6	28.6	8.0	30.2	45.8
	2	28.0	52.3	26.9	22.2	38.5	27.8
MDN_3	All	**39.0**	**57.1**	**42.1**	22.1	**43.0**	**50.1**
	1	12.7	31.9	8.2	15.7	20.0	6.6
	2	35.4	53.4	38.4	19.5	42.0	45.8
	3	22.2	34.7	22.6	2.9	20.9	39.8
MDN_5	All	38.8	56.8	41.7	**22.2**	43.0	49.8
	1	24.0	42.9	23.7	16.7	40.1	24.4
	2	1.8	5.2	1.0	4.3	0.4	0.0
	3	10.6	22.1	9.3	20.9	13.9	1.0
	4	27.3	40.8	28.6	6.4	30.8	45.3
	5	11.1	18.4	10.8	0.1	6.2	22.3

Table 1: Object detection evaluation. For mixture models, we evaluate with the full model and separately for each component (Comp.) when it is used to make all predictions. All models are trained on HG for 1X from scratch.

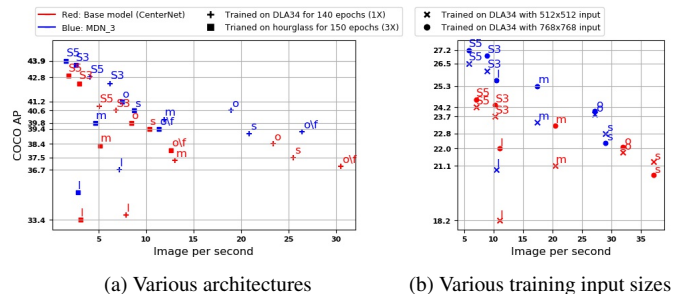(a) Various architectures   (b) Various training input sizes

Figure 3: Speed accuracy trade-off. **s**, **m**, **l**, and **o** indicate test time input size of 512, 768, 1024, and original input size, respectively. **S5** and **S3** indicate multi scale test with 5 and 3 scales, respectively. Due to limited resources for training on larger input size, models in (b) are trained for 100 epochs on 10% of coco-train (sampled uniformly at random.). Evaluations in (a) use left-right flip, except if marked with \f. Evaluations in (b) are done without left-right flip. Best seen in color and zoomed-in.
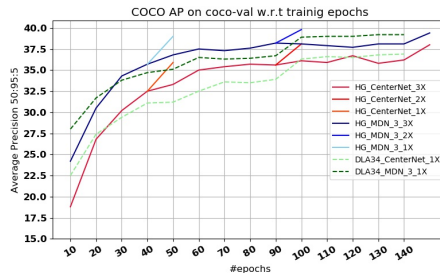

Figure 4: Convergence of MDN_3 vs. single-mode base.

question for future research; is it possible to achieve even higher accuracy with an increased number of components?

Model	FPS	AP	AP·50	AP·75	AP·S	AP·M	AP·L
YOLOv3 [34]	20	33.0	57.9	34.4	18.3	25.4	41.9
Gaussian YOLOv3 [7]	-	36.1	-	-	-	-	-
RetinaNet [25]	5.4	40.8	61.1	44.1	24.1	44.2	51.2
CornerNet [19]	4.1	40.5	56.5	43.1	19.4	42.7	53.9
ExtremeNet [45]	3.1	40.2	55.5	43.2	20.4	43.2	53.1
FSAF [46]	2.7	42.9	63.8	46.3	26.6	46.2	52.7
Single mode base on HG	8.44	0.401	0.589	0.436	0.232	0.435	0.496
MDN_3 (ours) on HG	7.47	0.416	0.599	0.452	0.239	0.451	0.518
Single mode base on DLA34	23.29	0.386	0.566	0.418	0.190	0.425	0.505
MDN_3 (ours) on DLA34	18.97	0.406	0.582	0.438	0.206	0.441	0.538

Table 2: Comparison to bottom-up state-of-the-art object detectors on *coco-test-dev*. Results correspond to single-scale inference with left-right flip augmentation at test time. Models on HG are trained for 3X and on DLA34 for 1X.

Model	Comp.	AP	AP·50	AP·75	AP·M	AP·L
Single mode base	-	46.4	75.8	50.3	43.9	53.2
MDN_2	All	52.3	78.2	57.2	50.0	58.9
	1	37.3	72.0	34.6	38.0	40.0
	2	37.7	62.6	39.1	33.4	46.2
MDN_3	All	52.3	77.2	58.0	50.6	59.3
	1	34.6	54.5	37.0	30.5	42.8
	2	40.7	70.9	42.0	39.0	46.8
	3	23.9	60.6	14.9	27.3	24.0

Table 3: Human pose estimation evaluation. For mixture models, we show the results for the full model and separately for each component when it is used to make all predictions. Models are trained on HG for 1X from scratch.

4.3.2 Speed-accuracy trade-off

In Fig. 3 we show the speed-accuracy trade-off for MDN_3 compared to the base single mode model. According to Fig. 3 (a), MDN_3 consistently achieves higher accuracy at higher speed. In particular, with 5 scale evaluation, MDN on DLA34 is as accurate as the CenterNet on stronger HG architecture, however twice faster (see red square labeled *S5* vs. blue cross labeled *S5*).

Given that MDN divides data based on object scale, we hypothesize that training on larger input should result in even better accuracy. So, on DLA34, we train Center-Net and MDN_3 on input size 768x768 (up from default 512x512). However, since training on larger input demands almost twice GPU memory, we train for 100 epochs on a 10% subset of *coco-train* sampled uniformly at random. This is solely due to the limited computational resources we have access to. Fig. 3 shows that (b), when trained on larger input, MDN_3 evaluated at single scale at resolution 768x768 surpasses accuracy of the base model evaluated with 5 scales, while being more than twice faster with FPS 17,47 vs. 7.04 (see red circle labeled *S5* vs. blue circle labeled *m*).

4.3.3 Convergence

In Fig. 4 we illustrated that an important aspect of using a mixture model is a faster convergence rate. For both HG and DLA34 architectures, MDN_3 gains higher accuracy more quickly than the single-mode base. The base model gradually recovers some of the gap, but never reaches the same accuracy level.

4.3.4 Comparison to the state-of-the-art

The essence of our contribution is to improve object detection by incorporating it into a mixture model. It is crucial to conduct experiments in a way that the real effect of the new formulation is easily understood. However, the official CenterNet model (the single-mode baseline) is trained for 50 epochs after initializing from ExtremeNet [45]. As we investigated, ExtremeNet itself is fine-tuned on top of CornerNet [19] for 50 epochs. And, CornerNet is trained from scratch for 100 epochs. It is safe to say that the model is trained for about 200 epochs (CenterNet4x). This schedule makes it difficult, if not impossible, to make a proper comparison and measure the effectiveness of the mixture formulation. Therefore, for comparisons on *coco-test-dev* too, we train both our mixture model and the base from scratch.

In table 2 we provide evaluation results along with run-time speed for our model and the single-stage state-of-the-art models on *coco-test-dev*. MDN_3 significantly improves the baseline model. Its precision is slightly lower than the recent FSAF model [46], however it is much faster. Please note that CenterNet4x achieves test AP of 42.1, slightly better than the AP of 41.5 achieved by MDN_3.

4.4. Human pose estimation

Table 3 shows the evaluation results for mixture models trained for human pose estimation on COCO. The mixture model leads by a significant margin in terms of AP. However, the interpretation of components is different than that of object detection. Here, only two significant modes are retrieved, no matter how many components we train. We experimented with up to 10 components and observed that having more than two components results in slightly better recall ($< 1\%$), but does not improve precision. By visualizing the predictions, it becomes clear that one of the modes focuses on roughly frontal view instances and the other one on instances with a backward view. Fig. 5b shows sample visualisation from MDN_3 on HG trained with 3X schedule. In the following, we provide more analysis regarding how the input space gets divided by the mixture model.

4.4.1 Analysis of the components

Gaussian mixture models enable us to analyze the model based on its uncertainty about samples and compute informative statistics of the modes. To this end, we analyze the difference among components of MDN_3 in terms of uncertainty. For the predictions made by each component, we compute the mean and standard deviation of the Gaussian variance terms, which quantifies uncertainty. The statistics, presented in table 4, show that indeed, there is a significant difference in the mean of the variance term for different components. We believe this difference is the actual

Component	Prediction rate	mean (std) σ_x	mean (std) σ_y	O_x^+ rate (Left)	O_y^+ rate (Left)	O_x^+ rate (Right)	O_y^+ rate (Right)
1 (Front)	73.1%	58 (52)	61 (57)	85%	51%	25%	50%
2 (-)	0.3%	803 (1106)	547 (846)	57%	51%	49%	55%
3 (Back)	26.6%	76 (119)	79 (120)	27%	48%	83%	47%

Table 4: For predictions made by each component of MDN_3 table shows the mean and standard deviation (std) of the variance terms in X and Y axes (σ_x and σ_y), and ratio of mean vectors (offset vectors) in the positive direction of X and Y axes (O_x^+ and O_y^+) for left and right parts.

reason behind dividing the data based on viewpoint. We also see that the second mode accounts for cases with more considerable uncertainty. But, it is responsible for only a negligible number of predictions.

Now let us look into some statistics of the mean vectors (i.e., offset vector) predicted by each mixture component. We divide the body parts into two sets of all the left and all the right parts. Then for each set, compute the portion of vectors in the positive direction of horizontal and vertical axes. We expect to see a significant difference in the direction of vectors along the horizontal axis between components and also between the left and right parts. Remember that vectors point from the body center to each keypoint, so if we go from the front view to the back view, the direction will be flipped in the horizontal axis. In table 4, we see a considerable difference in the horizontal direction of vectors for left and right parts between front and back components. This seems to be the reason for viewpoint turning out to be the dominant factor recovered by the mixture model.

We further compare the distribution of the samples in the dataset w.r.t. face visibility and occlusion of keypoints against the distribution of predictions by components of MDN_2. We use the nose visibility as a rough indicator of face visibility. As shown in table 5, the prediction distribution correlates well with the face visibility, which is an indicator of viewpoint in 2D. The majority of instances in the dataset are in frontal view, and similarly, the front view component makes the majority of the predictions. Related to our results, [2] have shown that excluding occluded keypoints from training leads to improved performance. More recently, [42] achieves more accurate 3D hand pose estimations by proposing a hierarchical model for estimation occlusion and using it to choose the right downstream model for making the final prediction.

4.4.2 Fine-grained evaluation

To understand what body parts gain the most from the MDN, we do a fine-grained evaluation for different subsets of keypoints. We modify the COCO evaluation script such that it only considers keypoints we are interested in. Table 6 shows the results. The first three rows of the table illustrate the sensitivity of the COCO evaluation metric. For the facial keypoints, where the metric is the most sensitive, the

Occluded Keypoints	Visible Keypoints	Occluded Face	Visible Face	Back mode	Front mode
12.3	87.7	30.3 (22.1)	69.7 (77.9)	25.5 (27.0)	74.5 (70.0)

Table 5: Distribution of ground truth face and keypoints visibility compared to that of maximum mixture component. Statistics are based on GT instances with more than 5 annotated keypoints (or ≥ 10), and model predictions with the score at least 0.5 (or ≥ 0.7). "visible face" includes instances with the visible and annotated nose, and "occluded face" those with the occluded or un-annotated nose.

	All	Facial	Non-Facial	Nose	Ears	Shoulders	Wrists	Hips	Ankles
GT Disp. 1	96.0	83.7	99.6	77.9	88.7	99.4	97.1	99.2	95.2
GT Disp. 2	80.4	47.4	93.4	42.3	59.8	93.4	86.8	97.2	91.7
GT Disp. 3	63.0	25.6	82.7	22.6	35.8	82.0	71.2	91.3	83.3
Base model	46.4	44.3	45.7	42.3	43.9	59.5	31.7	58.5	38.2
MDN_2	52.3	54.1	49.9	50.3	54.0	60.3	41.7	58.3	42.6

Table 6: Fine-grained evaluation. *GT Disp. x* means ground truth is displaced by *x* pixels in random direction.

improvement is larger. However, the biggest improvement comes for the wrists, which have the highest freedom to move. On the other hand, for torso keypoints, which are the most rigid, there is almost no improvement.

### 4.4.3 Comparison to the state-of-the-art

Table 7 compares our mixture model to the state-of-the-art on *coco-test-edv*. Analogous to our object detection setting, in order to make a proper comparison to CenterNet, we train it along with our mixture model from scratch with the 3X schedule on HG. The official CenterNet for human pose estimation is also trained with a 3X schedule but with different batch-size. Due to that, we observe a discrepancy in the results; we get an AP of 61.8 on the test server, but the official model gets 63. When no refinement is used, our training gets AP of 55.6 while the official model gets 55.0. Therefore, for the sake of fair comparison, in table 7, we only display the results obtained by our own training.

Model	FPS	AP	AP·50	AP·75	AP·M	AP·L
CMU-Pose [5]	12.98	61.8	84.9	67.5	58.0	70.4
AssociativeEmbedding [30]	9.3	62.8	84.6	69.2	57.5	70.6
PersonLab [32]	2.15	66.5	88.0	72.6	62.4	72.3
Single-mode base	7.65	55.6	82.8	61.1	49.6	65.9
MDN_3 (ours)	7.26	57.9	82.7	63.7	52.3	67.8
Single-mode base w/o flip	12.79	56.0	82.6	61.6	52.6	63.7
MDN_3 w/o flip (ours)	11.77	59.0	82.7	65.3	56.4	65.9
Single-mode base refined	7.13	61.8	85.4	68.0	57.4	70.5
MDN_3 refined (ours)	7.04	62.9	85.1	69.4	58.8	71.4

Table 7: Comparison to the state-of-the-art bottom-up human pose estimators on *coco-test-dev*. Evaluations are done at single-scale, and with left-right flip. MDN_3 and the base model are trained on HG for 3X.

5. Conclusion and future work

We propose a mixture formulation for spatial regression in dense 2D object detection and human pose estimation.
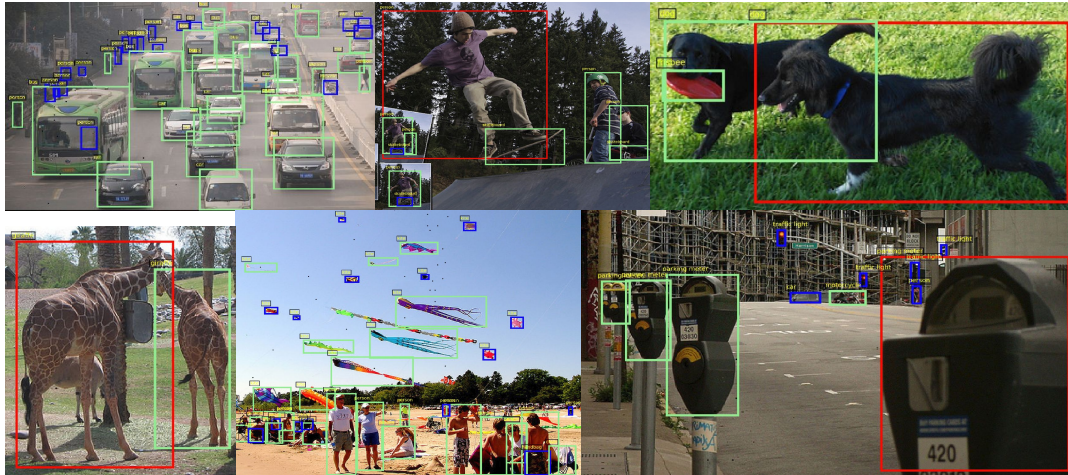
(a) Sample object detection by MDN_3. The three different modes are color-coded (blue, green, and red)

(b) Sample pose estimation by MDN_3. Ellipses represent uncertainty (σ_m in Eq. 4). The modes are color-coded using the bounding box color.

Figure 5: Sample predictions on *coco-val*. We provide more visualizations in the supplementary material.

We show that a mixture density network significantly improves accuracy on both tasks on a real-world large scale dataset. A mixture model provides much better speed-accuracy trade-off and can alleviate the need for multi-scale evaluation. Furthermore, it leads to faster convergence. For both object detection and human pose estimation, a mixture model splits data into meaningful modes; based on object scale and viewpoint, respectively. The models learn to choose the proper output head conditioned on input.

In human pose estimation, surprisingly, viewpoint is the dominant factor, and not the pose variation. This stresses that real-world data is multi-modal, but not necessarily in the way we expect. This also motivates using a more sophisticated pose representation in a single-mode model. Designing networks that are able to learn more diverse components is an exciting direction for further research.

Unlike most works on mixture models, here we use a very diverse large dataset without facing mode collapse. In the future, it will be valuable if one could provide an in-depth study of the role of size and diversity of data in the proper training of mixture models. Furthermore, it would be insight-full to build a density estimation model on more challenging tasks like the recent large vocabulary instance segmentation task (LVIS) [11], which has more than 1000 categories with huge data imbalance. Can mixture models learn to deal with fine modalities on such a diverse dataset?

Acknowledgments

References

[1] Yali Amit and Pedro Felzenszwalb. Object detection. *Computer Vision: A Reference Guide*, pages 537–542, 2014. 3

[2] Vasileios Belagiannis and Andrew Zisserman. Recurrent human pose estimation. In *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*, pages 468–475. IEEE, 2017. 7

[3] Maxim Berman, Amal Rannen Triki, and Matthew B Blaschko. The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4413–4421, 2018. 1

[4] Christopher M Bishop. Mixture density networks. 1994. 1, 2, 3

[5] 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. 2, 3, 7

[6] 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. 2

[7] Jiwoong Choi, Dayoung Chun, Hyun Kim, and Hyuk-Jae Lee. Gaussian yolov3: An accurate and fast object detector using localization uncertainty for autonomous driving. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. 6

[8] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015. 5

[9] Henggang Cui, Vladan Radosavljevic, Fang-Chieh Chou, Tsung-Han Lin, Thi Nguyen, Tzu-Kuo Huang, Jeff Schneider, and Nemanja Djuric. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2090–2096. IEEE, 2019. 4

[10] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 2

[11] Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5356–5364, 2019. 8

[12] Abner Guzman-Rivera, Dhruv Batra, and Pushmeet Kohli. Multiple choice learning: Learning to produce multiple structured outputs. In *Advances in Neural Information Processing Systems*, pages 1799–1807, 2012. 2

[13] 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. 2

[14] 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

[15] Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994. 1

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1, 4

[17] Sven Kreiss, Lorenzo Bertoni, and Alexandre Alahi. Pifpaf: Composite fields for human pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11977–11986, 2019. 3

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[19] 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. 4, 6

[20] Stefan Lee, Senthil Purushwalkam Shiva Prakash, Michael Cogswell, Viresh Ranjan, David Crandall, and Dhruv Batra. Stochastic multiple choice learning for training diverse deep ensembles. In *Advances in Neural Information Processing Systems*, pages 2119–2127, 2016. 2

[21] Stefan Lee, Senthil Purushwalkam, Michael Cogswell, David Crandall, and Dhruv Batra. Why m heads are better than one: Training a diverse ensemble of deep networks. *arXiv preprint arXiv:1511.06314*, 2015. 2

[22] Chen Li and Gim Hee Lee. Generating multiple hypotheses for 3d human pose estimation with mixture density network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9887–9895, 2019. 2, 4

[23] Wenbo Li, Zhicheng Wang, Binyi Yin, Qixiang Peng, Yuming Du, Tianzi Xiao, Gang Yu, Hongtao Lu, Yichen Wei, and Jian Sun. Rethinking on multi-stage networks for human pose estimation. *arXiv preprint arXiv:1901.00148*, 2019. 2

[24] Yanghao Li, Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. Scale-aware trident networks for object detection. *arXiv preprint arXiv:1901.01892*, 2019. 2

[25] 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. 1, 2, 6

[26] 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. 4

[27] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 2

[28] Osama Makansi, Eddy Ilg, Ozgun Cicek, and Thomas Brox. Overcoming limitations of mixture density networks: A sampling and fitting framework for multimodal future prediction.

In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7144–7153, 2019. 2

[29] Geoffrey J McLachlan and Kaye E Basford. *Mixture models: Inference and applications to clustering*, volume 84. M. Dekker New York, 1988. 1, 3

[30] Alejandro Newell, Zhiao Huang, and Jia Deng. Associative embedding: End-to-end learning for joint detection and grouping. In *Advances in Neural Information Processing Systems*, pages 2277–2287, 2017. 2, 7

[31] 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. 1, 2

[32] 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. 2, 3, 7

[33] Sergey Prokudin, Peter Gehler, and Sebastian Nowozin. Deep directional statistics: Pose estimation with uncertainty quantification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 534–551, 2018. 2

[34] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 2, 6

[35] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1, 2

[36] Christian Rupprecht, Iro Laina, Robert DiPietro, Maximilian Baust, Federico Tombari, Nassir Navab, and Gregory D Hager. Learning in an uncertain world: Representing ambiguity through multiple hypotheses. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3591–3600, 2017. 2, 4

[37] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1

[38] Bharat Singh, Mahyar Najibi, and Larry S Davis. Sniper: Efficient multi-scale training. In *Advances in Neural Information Processing Systems*, pages 9310–9320, 2018. 2

[39] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 1

[40] David H Wolpert. The lack of a priori distinctions between learning algorithms. *Neural computation*, 8(7):1341–1390, 1996. 1

[41] David H Wolpert, William G Macready, et al. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997. 1

[42] Qi Ye and Tae-Kyun Kim. Occlusion-aware hand pose estimation using hierarchical mixture density network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 801–817, 2018. 2, 7

[43] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2403–2412, 2018. 4

[44] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019. 2, 3, 4

[45] Xingyi Zhou, Jiacheng Zhuo, and Philipp Krahenbuhl. Bottom-up object detection by grouping extreme and center points. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 850–859, 2019. 6

[46] Chenchen Zhu, Yihui He, and Marios Savvides. Feature selective anchor-free module for single-shot object detection. *arXiv preprint arXiv:1903.00621*, 2019. 6