Stochastic Conditional Layers: Embodiments, Architecture, Training and Inference

Stochastic conditional layers are trainable parameters used to define a noise distribution, where the parameters depend on labels or categories present in the training dataset. The stochastic conditional layers are combined using selection layers for the task of inference, when the labels or categories of the inputs remain unknown. The stochastic conditional layers are used to obfuscate a given input, such that the inference accuracy is not compromised.

1 Architecture

We categorize the stochastic conditional layers based on their architecture. Any trainable network or layer(s) with trainable parameters can act as stochastic conditional layers. These include (but are not limited to) additive layers, convolutional layers, fully connected layers, recurrent layers like Long Short Term Memory (LSTM), Gated Recurrent Unit (GRU), transformer layers, etc. In this section we discuss stochastic conditional convolutional layers and stochastic conditional fully connected layers as physical incarnations of the stochastic conditional layers.

1.1 Stochastic Conditional Convolutional Layers

We discuss stochastic conditional convolutional layers (Sec. 1.1.3) by first laying the ground for convolutional layers (Sec. 1.1.1) and conditional convolutional layers (Sec. 1.1.2).

1.1.1 Convolutional Layers

Convolution (in the context of neural networks) is a linear mathematical operation where a kernel k slides across an input tensor x performing a linear operation at every location of the tensor x, thereby transforming x in a certain way. The output of this operation is a tensor h_k which represents a feature (also called an activation). In a convolutional layer of a neural network, the input tensor x is passed through a number of parameterized kernels, whose parameters are learnt during training through backpropagation. The activations h_k from the respective kernels k are stacked into channels to form the output $h = [h_k]$. Eq. (1) shows the convolution operation. In Eq. (1), $[m, n]$ represents the spatial coordinates of the output tensor h_k, $[i, j]$ represents the spatial coordinates of the kernel k.

$$h_k[m, n] = (x * k)[m, n] = \sum_i \sum_j k[i, j] x[m + i, n + j] \tag{1}$$

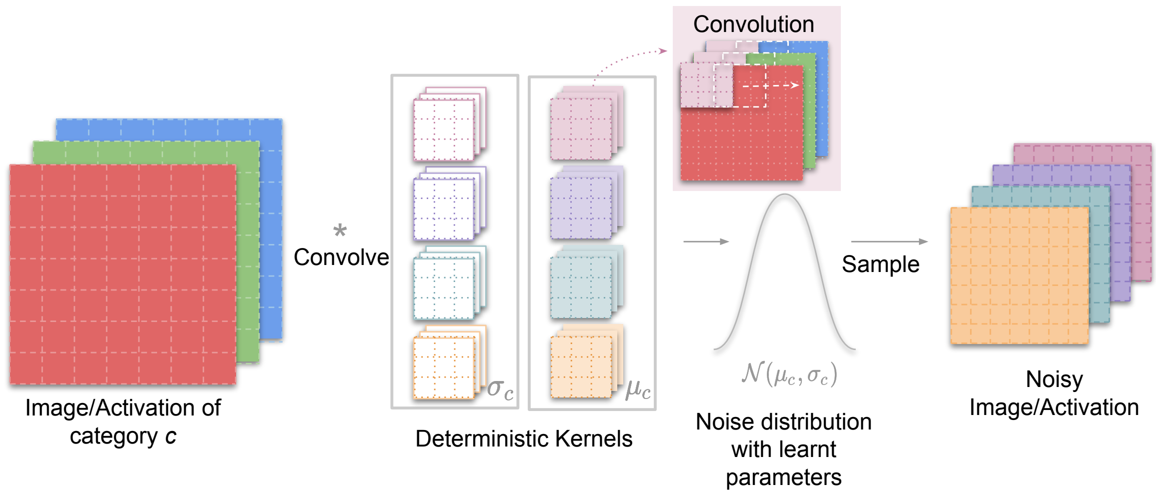
Image/Activation of category c
*
Convolve
σ_c
μ_c
Deterministic Kernels
Convolution
$\mathcal{N}(\mu_c, \sigma_c)$
Noise distribution with learnt parameters
Sample
Noisy Image/Activation

Figure 1: Stochastic conditional convolutional layer with learnt kernels μ_c and σ_c. The conditional probability distributions $\mathcal{N}(\mu_c, \sigma_c)$ are conditioned on the image category or label c.

1.1.2 Conditional Convolutional Layers

Deep networks are often employed for tasks that involve categorizing objects into a specific category from the training dataset. For example, object classification involves determining whether a given image is of a cat or a dog. Object detection involves the same, with the additional task of localizing the animal spatially. It can be useful to learn convolutional kernels specific to the category of objects. In other words, convolutional layers can be conditioned on the category of object. This is shown in Eq. (2), where k_c represents the kernels specific to the category c in the training dataset.

$$h_{k_c}[m,n] = (x * k_c)[m,n] = \sum_i \sum_j k_c[i,j] x[m+i, n+j] \tag{2}$$

1.1.3 (Our Innovation) Stochastic conditional convolutional layers

To introduce stochasticity, the output activations (h_{k_c} in Eq. (2)), obtained as a result of the convolution operation between input x and kernels k_c, acts as parameters of a probability distribution. Any probability distribution is applicable according to the use case including (but not limited to) gaussian, binomial and multinomial distributions. Fig. 1 shows the mechanism where the kernels convolve over an input to produce the parameters of a gaussian distribution, mean (μ_c) and standard deviation (σ_c), conditioned on the image category c. Instead of following the usual practice of using h_{k_c} in Eq. (2) directly as inputs to the next layers, we sample from the parameterized probability distribution and use this sample as an input to the next layer. This is elaborated in Sec. 3.

1.2 Stochastic Conditional Fully Connected Layers

We discuss stochastic conditional fully connected layers (Sec. 1.2.3) by first laying the ground for fully connected layers (Sec. 1.2.1) and conditional fully connected layers (Sec. 1.2.2).

1.2.1 Fully Connected Layers

A fully connected layer performs the inner-product between the input activation vector (x) and the trainable parameter vector W. This is represented by Eq. (3). The vector h represents the output activation that propagates forward.

$$h = W \cdot x \tag{3}$$

1.2.2 Conditional Fully Connected Layers

It is often useful to learn weights W_c specific to the category of objects given in the training dataset. In other words, fully connected layers can be conditioned on the category of object. This is shown in Eq. (4), where W_c represents the weights specific to the category c in the training dataset.

$$h_c = W_c \cdot x \tag{4}$$

1.2.3 (Our Innovation) Stochastic Conditional Fully Connected Layers

To introduce stochasticity, the output activations (h_c in Eq. (4)) obtained as a result of the inner product between the input x and weights W_c acts as parameters of a probability distribution. Any probability distribution is applicable according to the use case including (but not limited to) gaussian, binomial and multinomial distributions. Instead of using W_c in Eq. (4) directly as inputs to the next layers, we sample from the probabibility distribution and use the sample as an input to the next layer. This is elaborated in Sec. 3.

2 Applications

Stochastic conditional layers are useful for multi-class classification, multi-object detection, semantic and instance segmentation, multi-object tracking, and many more applications. We elaborate multi-class image classification (2.1) and multi-object detection (2.2) applications in this section. We also demonstrate how stochastic conditional layers can be used in the context of federated learning (2.3) for multi-class image classification. Note that stochastic conditional layers are applicable to other use-cases.

2.1 Multi-Class Image Classification

Multi-class image classification is the task of categorizing an image into one class or category, when the training dataset contains multiple categories. We can obfuscate a given image using different embodiments of stochastic conditional layers as described in Sec. 5.

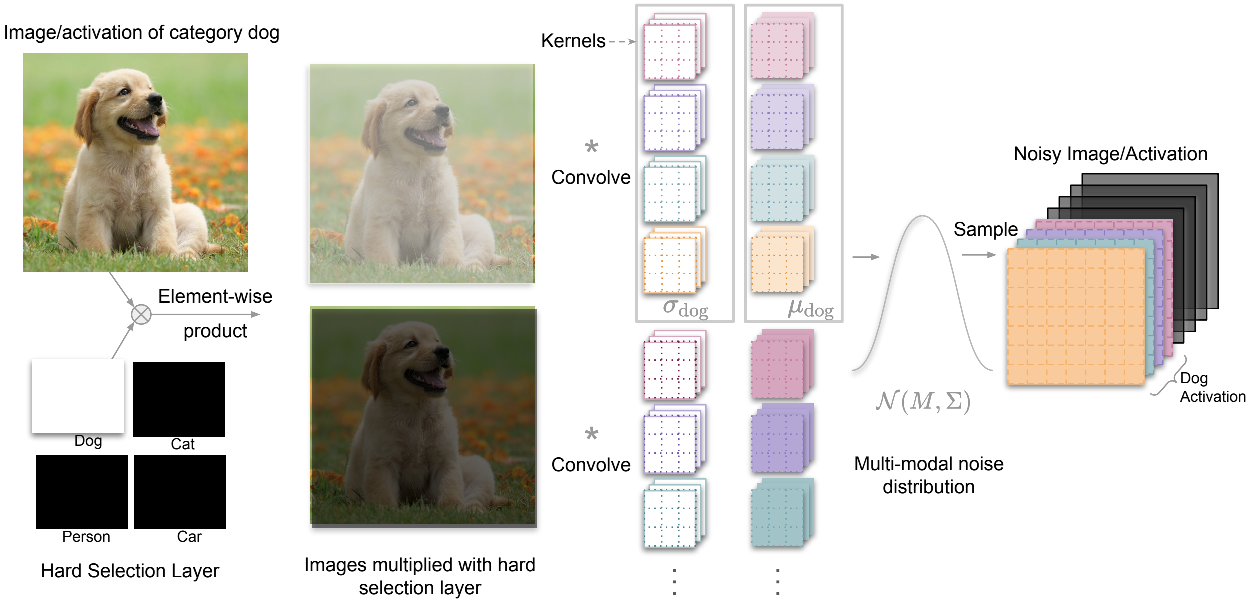

Figure 2: Stochastic conditional convolutional layer for multi-class classification. The multi-modal conditional probability distribution $\mathcal{N}(M, \Sigma)$, where $M = \{\mu_c\}$ and $\Sigma = \{\sigma_c\}$) is used to sample the output activation. μ_c and σ_c are conditioned on the image category or label c ($c =$ dog in this case). During training, the tensor π_{dog} in the hard selection layer is 1. All other tensors in the hard selection layer are 0. Learnt soft selection layer is used during inference.

The training and inference procedures of stochastic layers in the context of multi-class image classification are described in Sec. 3.

2.1.1 (Our Innovation) Selection Layer

To use stochastic conditional layers during inference, when the category of images (labels) are not known, we develop the idea of selection layers as shown in Fig. 2. Selection layers are used to combine the stochastic conditional layer that are designated for different labels. Each selection layer consists of C tensors, if there are C total categories in the training dataset. In one incarnation, each of the C tensors is element-wise multiplied with the input x before being convolved with all the convolutional kernels. In another incarnation, each of the C tensors is element-wise multiplied with the output obtained when the input x is convolved with all the convolutional kernels. We elaborate the use of selection layer below, with respect to the first incarnation described above.

Hard Selection Layer: During training of stochastic layers, the label for each image x is known. We can use a variant of the selection layer, called hard selection layer, where the C tensors have fixed pixel values 0 or 1 depending on the image category. The pixel values are 1 if the image matches the certain category. Otherwise, the pixel values are 0. This ensures that the given image of category c, only passes through kernels μ_c and σ_c, and no other sets of kernels, when x is element-wise multiplied with each tensor in the selection layer. The hard selection layer is shown in Fig. 2.

Soft Selection Layer: When the stochastic layers are fully trained using the hard selection layer described above, we make the selection layer trainable, and remove the constraint of 0 or 1 on the pixel values. The selection layer can have real pixel values between 0 and 1 and is called the soft selection layer. Each pixel value represents a probability that the particular region in the input is of interest for a category c. Freezing the trained stochastic layers, we train the parameters of the soft selection layer. This soft selection layer is used to combine the stochastic conditional layer during inference tasks, when the image category is not known.

2.2 Multi-Object Detection

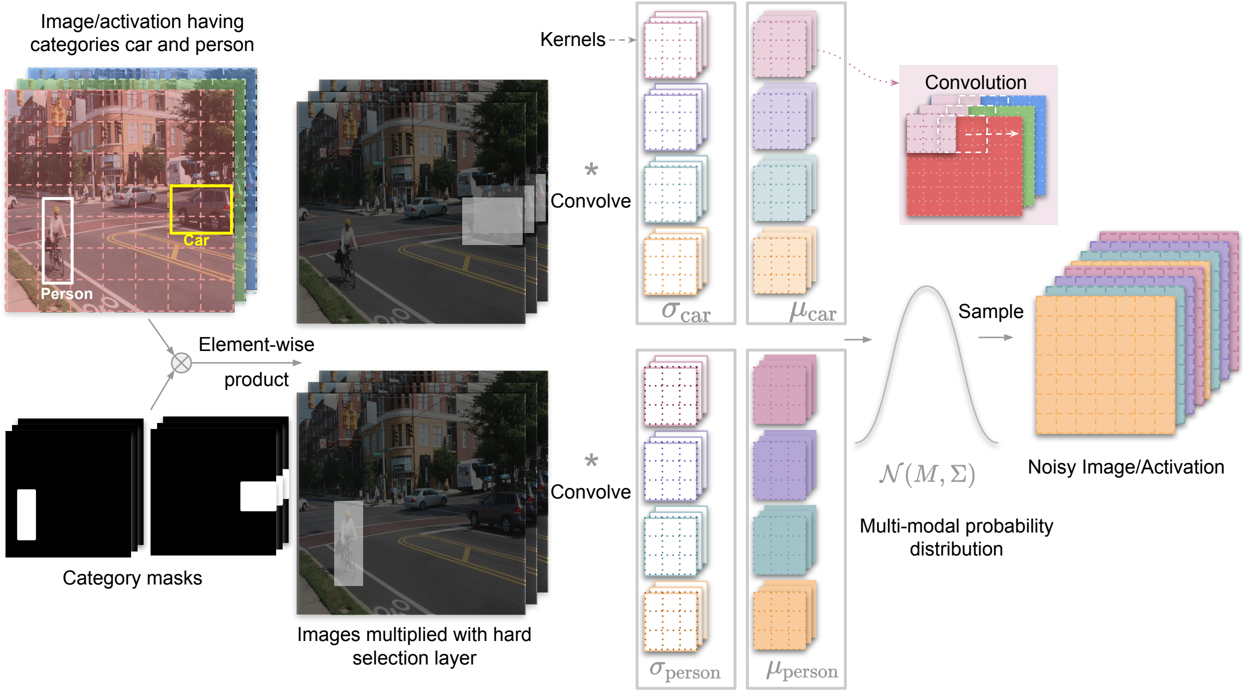

Figure 3: Stochastic conditional convolutional layer for multi-object detection. The multi-modal conditional probability distribution $\mathcal{N}(M,\Sigma)$, where $M = \{\mu_c\}$ and $\Sigma = \{\sigma_c\}$) is used to sample the output activation. μ_c and σ_c are conditioned on the image category or label c (c is person or car in this case). The tensor π_c in the hard selection layer have pixel values 0 or 1 during training, depending on the region of interest of category c. Trained soft selection layer is used during inference.

Multi-Object detection is the task of categorizing and localizing multiple objects, given an image. For example, objects of the categories "person" and "car" are to be detected in the input image shown in Fig. 3. We can obfuscate a given image using different embodiments of stochastic conditional layers as described in Sec. 5. The training and inference procedures of stochastic conditional layers in the context of multi-object detection is similar to Sec. 3.

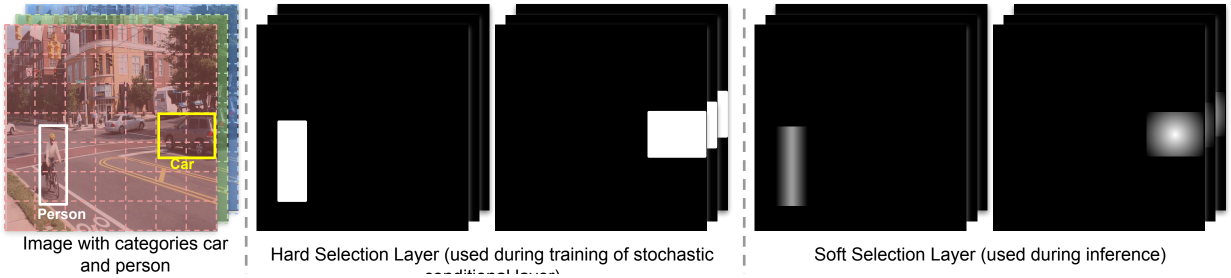
Person
Car
Image with categories car and person
Hard Selection Layer (used during training of stochastic
Soft Selection Layer (used during inference)

Figure 4: Selection layers used during training of stochastic layers and during inference. During training of stochastic layers, hard selection layer is used, where tensors have pixel values 0 or 1. Trained soft selection layer is used during inference, where pixel values are real and vary between 0 and 1, representing the probability that the particular pixel in tensor π_c belongs to class c.

2.2.1 (Our Innovation) Selection Layer

To use stochastic conditional layers during inference, when the category of objects in an images (labels) are not known, we develop the idea of selection layers as shown in Fig. 3. Selection layers are used to combine the stochastic conditional layer that are designated for different labels. Each selection layer consists of C tensors of the same size as input x, if there are C total categories in the training dataset. Each of the C tensors is element-wise multiplied with the input x before being convolved with all the convolutional kernels. We elaborate the use of selection layer below.

Hard Selection Layer: During training of stochastic layers, the label for each object in the image x is known. We can use a variant of the selection layer, called hard selection layer, where the C tensors have fixed pixel values 0 or 1 depending on the object category of the image. The value is 1 if the object matches the certain category. Otherwise, the value is 0. This ensures that the given object of category c, only passes through kernels μ_c and σ_c, and no other sets of kernels, when the input image x is element-wise multiplied with each tensor in the selection layer. The hard selection layer during the training of the stochastic conditional kernels is shown in Fig. 3 and Fig. 4 (middle figure). The white pixels indicate the value of 1, and black pixels indicate the value of 0. This ensures that the regions of interests are retained when the input image x is element-wise multiplied with each tensor in the selection layer.

Soft Selection Layer: When the stochastic layers are fully trained using the hard selection layer, we make the selection layer trainable, and remove the constraint on the selection layer to have fixed pixel values 0 or 1. The selection layer can have real pixel values between 0 and 1 and is called the soft selection layer. Each pixel represents a probability that the particular region is of interest for a certain category. Freezing the trained stochastic layers, we now train the parameters of the selection layer. The trained soft selection layer (Fig. 4 (right)) is used to combine the stochastic conditional layer during inference tasks, when the object categories are not known.

2.3 Federated Learning

We discuss learning of the stochastic noise layer in the federated learning setting. Any task like multi-class classification, multi-object detection, etc, can be considered for this. We discuss one embodiment of the federated learning procedure on the task of multi-class classification. Other embodiments follow similar procedure. Other embodiments, nonetheless, are not limited to federated learning.

1. We train a given neural network on 5% of the training dataset, instead of the entire dataset. Let us call this trained neural network NN-5.

2. We use NN-5 to train the stochastic conditional layer described earlier. We call this stochastic layer SL-5.

3. We already have 5% of the pure dataset. We obfuscate the rest 95% of the training data using SL-5 and regulariztion technique described in Sec. 2.3.1.

4. We then resume training of NN-5 on this 5% pure and 95% noisy dataset (obtained from the previous step). Let's call this trained neural network NN'-5.

5. We gradually make more training data available (10%, 15%, etc) and iterate over steps 1,2,and 3, until the desired accuracy is reached.

2.3.1 (Our Innovation) Regularization

In step 3 of the federated learning embodiment described above, we obfuscate 95% of the training data using a stochastic layer training on only 5% of the pure training dataset (SL-5). Note that this noisy data coming from the stochastic layer will be highly biased towards the small ratio of the data SL-5 was trained on. To reduce this bias, we propose a regularization step where we randomly screen parts of the noisy data at every iteration of the training, so that the screened parts aren't visible to the neural network.

3 Training and Inference of Stochastic Conditional Layers

In this section, we elaborate the training and inference procedures for the stochastic conditional layers, considering the convolutional incarnation and by assuming a gaussian distribution for the output activations for the task of multi-class classification. Other physical incarnations, types of distributions and applications will have similar procedures. We describe several applications in Sec. 2

3.1 Training

We describe training as a two step procedure. In the first step, we learn the weights of the stochastic conditional layer. In the next step, we learn the weights of the soft selection layers (described in Sec. 2.1) that are necessary during inference. The steps are described below. Note that the training pipeline can be different depending on applications. For example, step 2 can be omitted if the user is aware of the class labels. Hard Selection layers can be used in that case (Sec. 2.1). Fig. 2 describes a forward pass during the training procedure for the task of multi-class image classification.

3.1.1 Training of Stochastic Conditional Layer

Prior to training, two sets of kernels (k_{μ_c} and k_{σ_c}) are initialized for the two parameters, in a gaussian distrubution, mean (μ_c) and standard deviation (σ_c) respectively, conditioned on each category c out of the total C categories in the training dataset. During the forward pass, the kernels k_{μ_c} and k_{σ_c} perform convolution operation on the input activation x, if x belongs to the category c. In other words, x is first multiplied with all the C tensors in the hard selection layer, π_c, which have pixel values 1 if x belongs to category c, and 0 if x belongs to any other category. This modified input is then passed through the stochastic conditional layer. The output activation maps (μ_c and σ_c) are obtained from the respective set of kernels according to Eq. (5) and Eq. (6). μ_c and σ_c (mean and standard deviation) are used to define the gaussian distribution. We randomly sample an activation map h_c from this distribution according to Eq. (7). h_c acts as an input activation for the next layers in the network.

$$\mu_c[m,n] = (x * k_{\mu_c})[m,n] = \sum_i \sum_j k_{\mu_c}[i,j]x[m+i,n+j] \tag{5}$$

$$\sigma_c[m,n] = (x * k_{\sigma_c})[m,n] = \sum_i \sum_j k_{\sigma_c}[i,j]x[m+i,n+j] \tag{6}$$

$$\begin{aligned} h_c &\sim \mathcal{N}(\mu_c, \sigma_c) \\ \Rightarrow h_c &= \mu_c + \sigma_c.\epsilon;\ \epsilon \sim \mathcal{N}(0,1) \end{aligned} \tag{7}$$

The kernels are trained in a similar manner as standard convolutional neural networks. The parameters of the gaussian distribution for each category μ_c and σ_c (Eq. (5) and Eq. (6)), obtained in the forward pass, are differentiable with respect to the kernels k_{μ_c} and k_{σ_c} respectively. h_c is differentiable with respect to μ_c and σ_c. Gradients of the output activation h_c can be obtained with respect to the kernels k_{μ_c} and k_{σ_c}. The kernels are trainable using the aforementioned gradients through back-propagation and gradient descent.

3.1.2 Training of Soft Selection Layer

The selection layer can directly be applied to the input x. As other embodiments, the selection layer can also be applied after the stochastic layers, or anywhere in the neural network. We describe the training procedure considering that the selection layer is applied directly on the input.

The input x is multiplied with trainable tensor of the soft selection layer π_c whose values are real and vary between 0 and 1, representing the probability of the image belonging to a certain category c. If there are C categories in the dataset, there are C tensors in the soft selection layer to be trained. $\pi = \{\pi_c\}$ represents all the tensors concatenated together. The modified input $(x \bigotimes \pi)$ undergoes forward pass. The activations at every step are differentiable with respect to the tensors in the soft selection layer and hence backpropagation and gradient descent are directly applicable to train the soft selection layer. The step involving training of the soft selection layer can be omitted if the user is aware of the input category when the input is passed through the stochastic conditional layer. In that case, we can use the hard selection layer, where $\pi_c = 1$ if x belongs to category c, otherwise $\pi_c = 0$.

3.2 Inference

The kernels k_{μ_c} and k_{μ_c} and tensors in the soft selection layer, π_c, are trained according to the previous sub-section. A forward pass is performed using the category masks and trained kernels to produce the a parameterized probability distribution, from which the output activation map h_c is sampled. The output activation map acts as an input activation for the next layer in the neural network.

4 Related Work

Previous works attempted to ensure that the accuracy of the deep networks trained on very little carefully crafted or chosen training data, remain similar to the accuracy of the deep networks trained on full training datasets. Some of these attempts are described next.

Dataset distillation: Wang et al. [5] learned a small number of data points that do not need to come from the correct data distribution. A model trained on these synthetic data points approximate a model trained on the original data well.

Active learning: Active learning [2, 3, 4, 1] attempts to maximize the performance of models while annotating very little training data. The deep model is first trained using a small set of annotated training data. The model queries a user to label selected unlabelled samples for which the model had the lowest prediction confidence.

5 Embodiments of Stochastic Conditional Layers

We consider a neural network N, input x, a stochastic conditional layer S_L, and n additional regular layers $L_{i\{i\,=\,0\text{ to }n\}}$. In the normal state of using N, the input is applied to N without the involvement of S_L or L_i. We propose a few embodiments as described below.

1. In one embodiment, we propose this new arrangement as shown. $x \rightarrow L_{i\{i\,=\,0\text{ to }n\}} \rightarrow S_L \rightarrow N$.

2. In another embodiment, we break N into two parts N_1 and N_2, where N is equivalent to N_1,N_2 back to back. Let O_1 be the output of applying N_1 to x, i.e., $O_1 = x \rightarrow N_1$. Let O_2 be the output of applying $L_{i\{i\,=\,0\text{ to }n\}}$ to x, i.e., $O_2 = x \rightarrow L_{i\{i\,=\,0\text{ to }n\}}$.

Then, we merge O_1 and O_2 and pass the merged results through S_L. We then pass the resulting activations through N_2.

3. In another embodiment, we apply S_L to O_2, then merge the results with O_1, and then pass the result through N_2.

References

[1] Hamed H Aghdam, Abel Gonzalez-Garcia, Joost van de Weijer, and Antonio M López. Active learning for deep detection neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3672–3680, 2019.

[2] Xuefeng Du, Dexing Zhong, and Huikai Shao. Building an active palmprint recognition system. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 1685–1689. IEEE, 2019.

[3] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *International Conference on Machine Learning*, pages 1183–1192. PMLR, 2017.

[4] Denis Gudovskiy, Alec Hodgkinson, Takuya Yamaguchi, and Sotaro Tsukizawa. Deep active learning for biased datasets via fisher kernel self-supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9041–9049, 2020.

[5] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. Dataset distillation. *arXiv preprint arXiv:1811.10959*, 2018.