Stochastic Layers

1 Architecture

We categorize the stochastic layers based on their architecture. Any trainable network or a layer with trainable parameters can act as a stochastic layer. These include (but are not limited to) convolutional layers, fully connected layers, support vector machines, recurrent networks like LSTMs, transformers, etc. In this section we discuss convolutional stochastic layers and fully connected stochastic layers as physical incarnations of the stochastic layers.

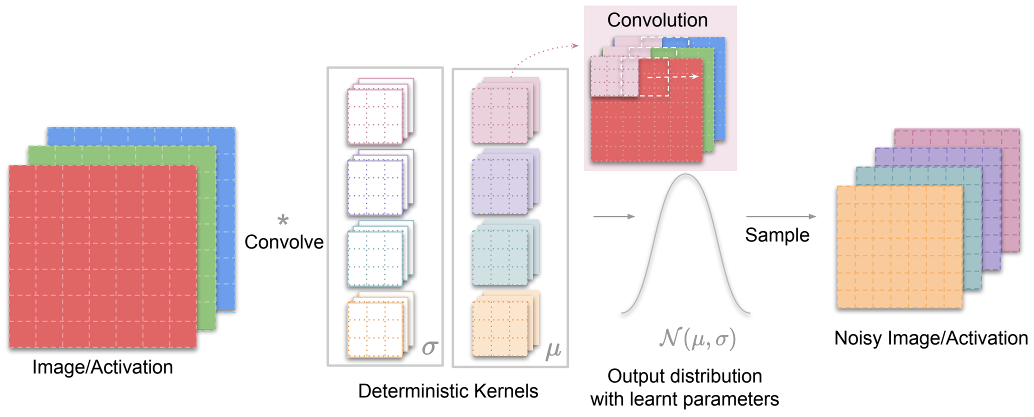

Figure 1: Stochastic layer with deterministic convolutional kernels.

1.1 Convolutional Stochastic layer

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, thereby transforming the tensor 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 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.

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

We have the following two proposed mechanisms for learning the kernels in the convolutional stochastic layers.

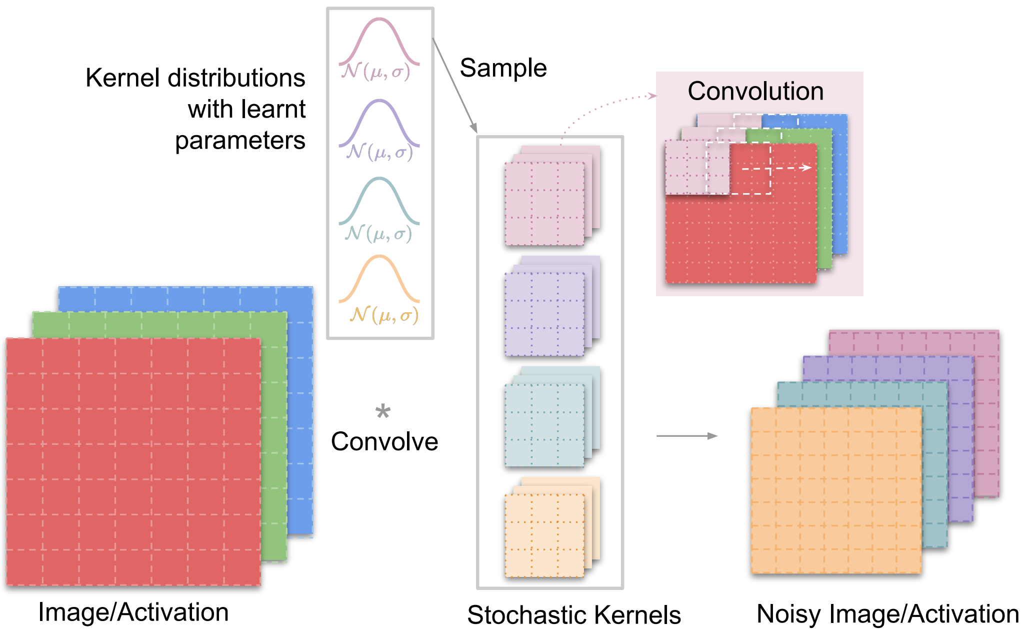
Kernel distributions with learnt parameters
$\mathcal{N}(\mu,\sigma)$
$\mathcal{N}(\mu,\sigma)$
$\mathcal{N}(\mu,\sigma)$
$\mathcal{N}(\mu,\sigma)$
Sample
Convolution
*
Convolve
Image/Activation
Stochastic Kernels
Noisy Image/Activation

Figure 2: Stochastic layer with stochastic convolutional kernels.

1.1.1 Deterministic Kernels for Convolutional Stochastic Layers

The deterministic kernels are used to produce outputs which act 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. Next, we elaborate the forward pass, training procedure and inference by assuming a gaussian distribution for the output activations.

Forward Pass: Prior to training, two sets of kernels (k_μ and k_σ) are initialized for the two parameters in a gaussian distrubution μ and σ respectively. During the forward pass, the kernels k_μ and k_σ perform convolution operation on the input activation x. The output activation maps (μ and σ) are obtained from the respective set of kernels according to Eq. (2) and Eq. (3). μ and σ are used to define the gaussian distribution. We randomly sample an activation map h_{sample} from this distribution according to Eq. (4). h_{sample} acts as an input activation for the next layers in the network.

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

$$\sigma[m,n] = (x * k_\sigma)[m,n] = \sum_i \sum_j k_\sigma[i,j] x[m+i, n+j] \tag{3}$$

$$\begin{aligned} h_{\text{sample}} &\sim \mathcal{N}(\mu, \sigma) \\ \Rightarrow h_{\text{sample}} &= \mu + \sigma.\epsilon;\ \epsilon \sim \mathcal{N}(0,1) \end{aligned} \tag{4}$$

Training: The deterministic kernels are trained in a similar manner as standard convolutional neural networks. The parameters of the gaussian distribution μ and σ (Eq. (2) and Eq. (3)), obtained in the forward pass, are differentiable with respect to the kernels k_μ and k_σ respectively. h_{sample} is differentiable with respect to μ and σ. Gradients of the output activation h_{sample} can be obtained with respect to the kernels k_μ and k_σ. The kernels are trainable using the aforementioned gradients through back-propagation and gradient descent.

Inference: The kernels k_μ and k_σ are trained according to the previous sub-section. A forward pass is performed using the trained kernels to produce the output activation map h_{sample}. The output activation map acts as an input activation for the next layer in the neural network.

1.1.2 Stochastic Kernels for Convolutional Stochastic Layers

In contrast to single-valued kernels used in standard deep networks, we can introduce stochasticity by learning a probability distribution representing each kernel. The stochastic kernels are drawn from the respective probability distributions, according to Eq. (6). The parameters of the probability distributions are trainable. Any probability distribution is applicable for a kernel according to the use case including (but not limited to) gaussian, binomial and multinomial distributions. Fig. 2 shows the mechanism where the kernels, sampled from respective gaussian distributions, convolve over an input to produce an output activation map. Next, we elaborate the forward pass, training procedure and inference by assuming gaussian distributions for the kernels.

Forward Pass: Prior to training, the parameters ($\mu_{k_{\text{stoc.}}}$, $\sigma_{k_{\text{stoc.}}}$), representing a gaussian for each kernel distribution, are initialized. During forward pass, a kernel $k_{\text{stoc.}}$ is drawn from each distribution according to Eq. (6). $k_{\text{stoc.}}$ convolves over the input activation x to produce the output activation ($h_{k_{\text{stoc.}}}$) (Eq. (5)).

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

$$\begin{aligned} k_{\text{stoc.}} &\sim \mathcal{N}(\mu_{k_{\text{stoc.}}}, \sigma_{k_{\text{stoc.}}}) \\ \Rightarrow k_{\text{stoc.}} &= \mu_{k_{\text{stoc.}}} + \sigma_{k_{\text{stoc.}}}.\epsilon;\ \epsilon \sim \mathcal{N}(0,1) \end{aligned} \tag{6}$$

Training: The parameters of the gaussian distribution ($\mu_{k_{\text{stoc.}}}$, $\sigma_{k_{\text{stoc.}}}$) for each kernel k are trainable. The output activation $h_{k_{\text{stoc.}}}$ is differentiable with respect to the kernels $k_{\text{stoc.}}$. The kernels $k_{\text{stoc.}}$ are differentiable with respect to parameters ($\mu_{k_{\text{stoc.}}}$, $\sigma_{k_{\text{stoc.}}}$). Gradients of the output activation $h_{k_{\text{stoc.}}}$ can be obtained with respect to the parameters ($\mu_{k_{\text{stoc.}}}$, $\sigma_{k_{\text{stoc.}}}$). Back-propagation and gradient descent are directly applicable.

Inference: The parameters ($\mu_{k_{\text{stoc.}}}$, $\sigma_{k_{\text{stoc.}}}$), representing each kernel distribution, are trained following the previous sub-section. A forward pass is performed using the trained parameters to produce the output activation map $h_{k_{\text{stoc.}}}$which acts as an input activation for the next layer in the neural network.

1.2 Fully Connected Stochastic Layer

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. (7). The vector h represents the output activation that propagates forward.

$$h = W \cdot x \tag{7}$$

1.2.1 Deterministic Weight Vector for Fully Connected Stochastic Layers

The deterministic weight vectors are used to produce output activations which act 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. We discuss the forward pass, training and inference procedure, by assuming a gaussian probability distribution for the output activation.

Forward Pass: Prior to training, two sets of weight vectors (W_μ and W_σ) are initialized for the two parameters μ and σ. During each forward pass, inner product between the input activation x and the weight vectors W_μ and W_σ produces the output activation maps μ and σ according to Eq. (8) and Eq. (9). μ and σ are used to define the gaussian distribution. We randomly sample an activation map h_{sample} from this distribution according to Eq. (10). h_{sample} acts as an input activation for the following layers in the network.

$$\mu = W_\mu \cdot x \tag{8}$$

$$\sigma = W_\sigma \cdot x \tag{9}$$

$$\begin{gathered} h_{\text{sample}} \sim \mathcal{N}(\mu, \sigma) \\ \Rightarrow h_{\text{sample}} = \mu + \sigma.\epsilon;\ \epsilon \sim \mathcal{N}(0, 1) \end{gathered} \tag{10}$$

Training: The deterministic weight vectors are trained in a similar manner as standard fully connected layers in a neural networks. The parameters of the gaussian distribution μ and σ (Eq. (8) and Eq. (9)), from the forward pass, are differentiable with respect to the weight vectors W_μ and W_σ respectively. h_{sample} is differentiable with respect to μ and σ. Gradients of the output activation h_{sample} can be obtained with respect to W_μ and W_σ. The weights are trainable using back-propagation and gradient descent.

Inference: The weight vectors W_μ and W_σ are trained according to the previous sub-section. A forward pass is performed using the trained weight vectors to produce the output activation map h_{sample}. The output activation map acts as an input activation for the next layer in the neural network.

1.2.2 Stochastic Weight Vector for Fully Connected Stochastic Layers

In contrast to single-valued weight vectors used in standard deep networks, we can introduce stochasticity by learning a probability distribution representing the weights. The stochastic weight vectors are drawn from the learnt probability distribution, according to Eq. (12). The parameters of the probability distribution are trainable. Any probability distribution is applicable according to the use case including (but not limited to) gaussian, binomial and multinomial distributions. Next, we discuss the forward pass, training procedure and inference by assuming the trainable probability distribution for the weight vector to be a gaussian.

Forward Pass: Prior to training, the parameters ($\mu_{W_{\text{stoc.}}}$, $\sigma_{W_{\text{stoc.}}}$) representing a gaussian are initialized. During forward pass, a weight vector is sampled from each distribution according to Eq. (12). Inner product of the input activation x and $W_{\text{stoc.}}$ produces the output activation ($h_{W_{\text{stoc.}}}$) according to Eq. (11).

$$h_{W_{\text{stoc.}}} = W_{\text{stoc.}} \cdot x \tag{11}$$

$$\begin{aligned} W_{\text{stoc.}} &\sim \mathcal{N}(\mu_{W_{\text{stoc.}}}, \sigma_{W_{\text{stoc.}}}) \\ \Rightarrow W_{\text{stoc.}} &= \mu_{W_{\text{stoc.}}} + \sigma_{W_{\text{stoc.}}}.\epsilon;\ \epsilon \sim \mathcal{N}(0,1) \end{aligned} \tag{12}$$

Training: The parameters of the gaussian distribution ($\mu_{W_{\text{stoc.}}}$, $\sigma_{W_{\text{stoc.}}}$) are trainable. The output activation $h_{W_{\text{stoc.}}}$ is differentiable with respect to $W_{\text{stoc.}}$. The weight vector $W_{\text{stoc.}}$ is differentiable with respect to parameters ($\mu_{W_{\text{stoc.}}}$, $\sigma_{W_{\text{stoc.}}}$). Gradients of the output activation $h_{W_{\text{stoc.}}}$ can be obtained with respect to ($\mu_{W_{\text{stoc.}}}$, $\sigma_{W_{\text{stoc.}}}$). Back-propagation and gradient descent are directly applicable.

Inference: The parameters ($\mu_{W_{\text{stoc.}}}$, $\sigma_{W_{\text{stoc.}}}$), representing a distribution for the weight vector, are trained following the previous sub-section. A forward pass is performed using the trained parameters to produce the output activation map $h_{W_{\text{stoc.}}}$ which acts as an input activation for the next layer in the neural network.

2 Embodiments of Stochastic Layers

We consider a neural network N, input x, a stochastic 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 descried 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.