Convolutional Networks

Instructor: John Thickstun

Discussion Board: Available on Ed

Zoom Link: Available on Canvas

Instructor Contact: thickstn@cs.washington.edu

Course Webpage: https://courses.cs.washington.edu/courses/cse599i/20au/

Recap: Variational Autoencoders

- Generative model $p_\theta(x,z) = p_\theta(x|z)r(z)$. Learn $p_\theta(x) \approx p(x)$, where

$$p_\theta(x) = \mathbb{E}_{z\sim r}[p_\theta(x|z)] = \int_{\mathcal{Z}} p_\theta(x|z)r(z)\,dz.$$

- Estimate the MLE using the ELBO:

$$\hat{\theta}_{\text{mle}} \equiv \arg\max_\theta \mathop{\mathbb{E}}_{x\sim p} \log p_\theta(x) = \arg\max_\theta \sup_{q_\varphi} \mathop{\mathbb{E}}_{\substack{x\sim p \\ z\sim q_\varphi(\cdot|x)}} \log \frac{p_\theta(x,z)}{q_\varphi(z|x)}.$$

- Modeling choices: prior $r(z)$, likelihood $p_\theta(x|z)$, and proposal $q_\varphi(z|x)$.

Recap: The Gaussian VAE

- Use a prior $r(z) = \mathcal{N}(0, I_{\mathcal{Z}})$.
- Gaussian likelihood $p_\theta(x|z) = \mathcal{N}(x; g_\theta(z), \sigma_\theta^2(z)I)$.
- Decoder $x = g_\theta(z) + \sigma_\theta(z)\varepsilon_x, \text{where } \varepsilon_x \sim \mathcal{N}(0, I_{\mathcal{X}})$.
- Gaussian posterior approximation $q_\varphi(z|x) = \mathcal{N}(z; f_\varphi(x), \Sigma_\varphi(x))$.
- Encoder $z = f_\varphi(x) + \Sigma_\varphi^{1/2}(x)\varepsilon_z, \text{where } \varepsilon_z \sim \mathcal{N}(0, I_{\mathcal{Z}})$.
- How to parameterize the neural networks? What is the structure of the data?

Image Modeling

- Represent an image $\mathbf{x}$ as a tensor $\mathbf{x} \in \mathbb{R}^{C \times w \times h}$.

  ➡ Color channels C, height h, width w.

- For R/G/B images, C = 3; for grayscale C = 1.
- Normalize color intensities $\mathbf{x}_{c,i,j} \in [0, 1]$.
- Color intensities at position (i,j) constitute a pixel.
- Can discretize intensity values, e.g. 8-bit. color.

Vahdat and Kautz (Preprint 2020)

Convolutional Networks

- A **convolutional layer** is a family of functions $b : \mathbb{R}^{C\times d\times d} \to \mathbb{R}^{C\times d\times d}$.

- A **convolutional neural network** (**convnet, CNN**) is a stack of blocks:

$$f = b^{(L)} \circ \cdots \circ b^{(1)} : \mathbb{R}^{C\times d\times d} \to \mathbb{R}^{C\times d\times d}.$$

- Compare this spatial transformation to the sequential transformer model.

- 2-d spatial structure isn't so important: generalizes to 1-d or 3-d convnet.

Convolutional Layer

- A **convolutional layer** is a family of functions $f_\theta : \mathbb{R}^{C_{\text{in}} \times d \times d} \to \mathbb{R}^{C_{\text{out}} \times d \times d}$.

- If $\mathbf{x} \in \mathbb{R}^{C_{\text{in}} \times d \times d}$ then $f_\theta(\mathbf{x}) = \mathbf{z}$ where

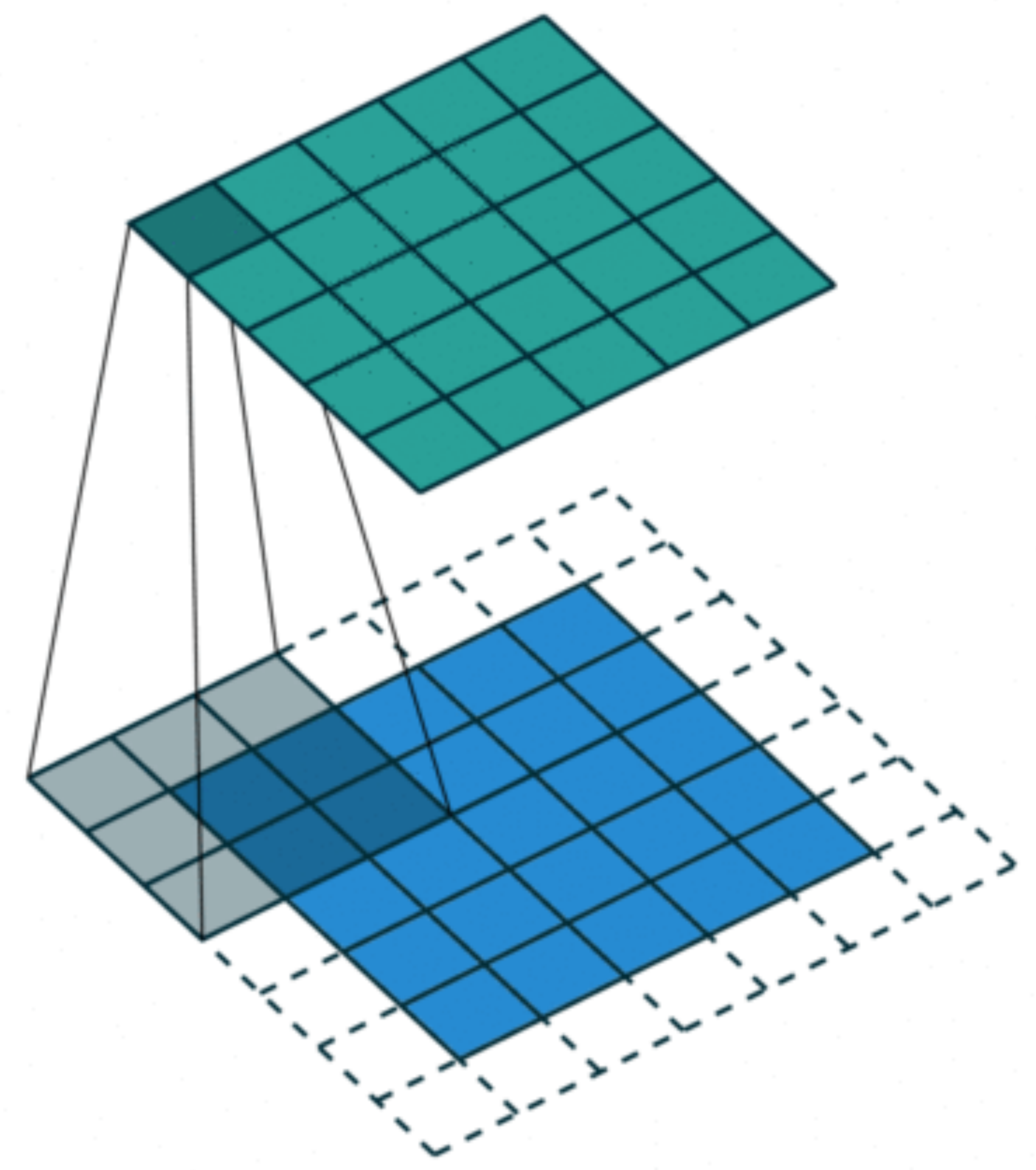
$$\mathbf{z}_{c,i,j} = \text{ReLU}\left(\sum_{c'=1}^{C_{\text{in}}} \langle W_{c,c'}, \text{Pad}(\mathbf{x})_{i:i+k,j:j+k} \rangle\right), \qquad W \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}} \times k \times k}.$$

- And the Pad function is defined by

$$\text{Pad}(\mathbf{x})_{d,i,j} = \begin{cases} \mathbf{x}_{d,i-\lfloor k/2 \rfloor, j-\lfloor k/2 \rfloor} & \text{if } i \geq \lfloor k/2 \rfloor \text{ and } j \geq \lfloor k/2 \rfloor, \\ 0 & \text{otherwise.} \end{cases}$$

Convolutional Layer

- Visualizing application of a filter W_c to a single input channel $\mathbf{x}_c$.
- Zero-padding ensures that the output $\mathbf{z}$ has the same dimensions as the input $\mathbf{x}$.
- Compare this spatial transformation to the sequential transformer model.
- Easy generalization to 1-d or 3-d convnet.



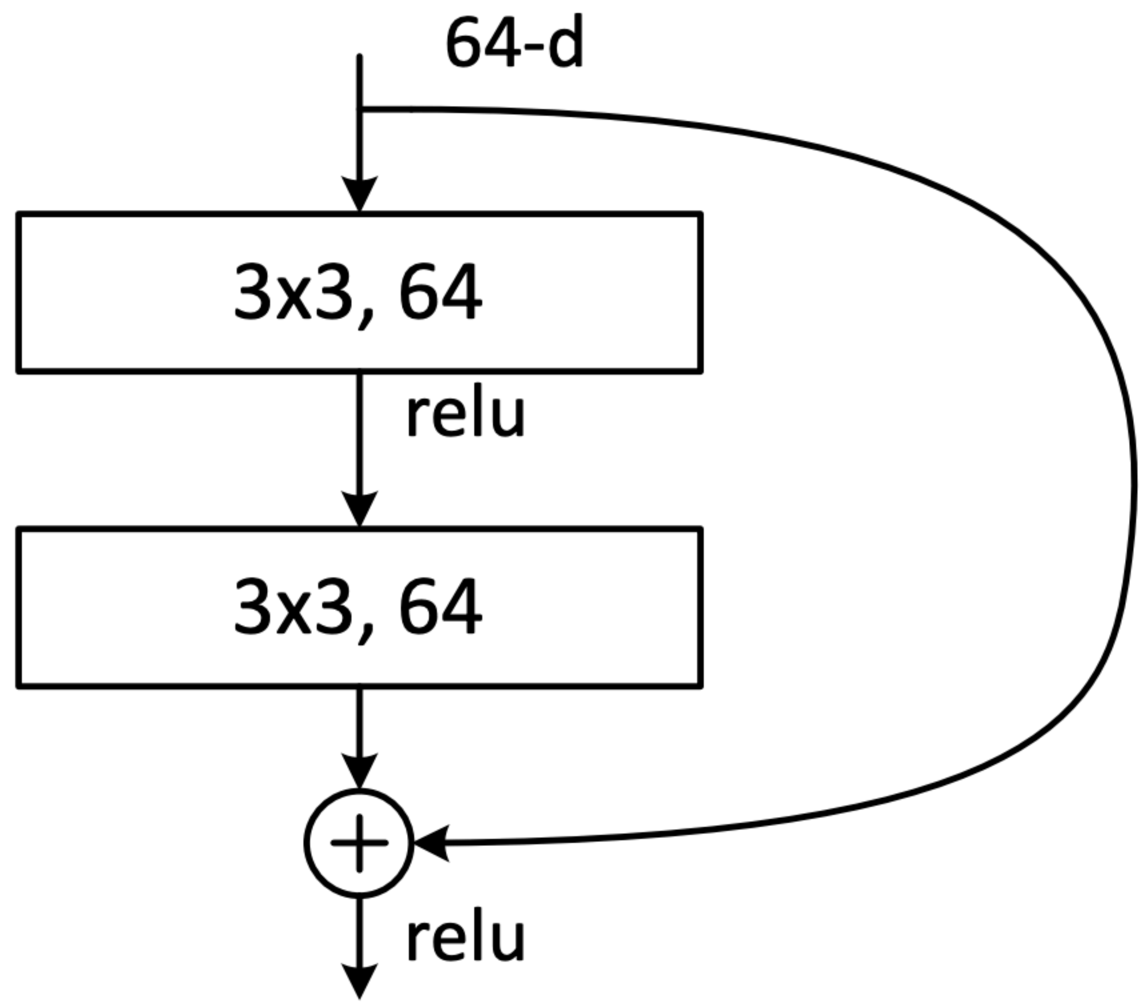
Dumoulin and Visin, 2019

Deep Residual Convnets

- A **residual convolutional block** is a family $f_\theta : \mathbb{R}^{C_{\text{in}} \times d \times d} \to \mathbb{R}^{C_{\text{out}} \times d \times d}$.
- If $\mathbf{x} \in \mathbb{R}^{C \times d \times d}$ then $f_\theta(\mathbf{x}) = \mathbf{z}$ where

Transformation	Weights
$\mathbf{u}'_{c,i,j} = \sum_{c'=1}^{C} \langle W^1_{c,c'}, \text{Pad}(\mathbf{x})_{i:i+k,j:j+k} \rangle,$	$W^1 \in \mathbb{R}^{C \times C \times k \times k},$
$\mathbf{u} = \text{ReLU}\left(\text{BatchNorm}(\mathbf{u}'; \gamma_1, \beta_1)\right),$	$\gamma_1, \beta_1 \in \mathbb{R}^C,$
$\mathbf{z}'_{c,i,j} = \sum_{c'=1}^{C} \langle W^2_{c,c'}, \text{Pad}(\mathbf{u})_{i:i+k,j:j+k} \rangle,$	$W^2 \in \mathbb{R}^{C \times C \times k \times k}.$
$\mathbf{z} = \text{ReLU}\left(\mathbf{x} + \text{BatchNorm}(\mathbf{z}'); \gamma_2, \beta_2\right),$	$\gamma_2, \beta_2 \in \mathbb{R}^{\mathbb{C}}.$

(C = 64, k = 3)



He et. al., CVPR 2016

Batch Normalization

Transformation	Weights
$\mathrm{BatchNorm}(\mathbf{z};\gamma,\beta)_{i,c} = \gamma_c \frac{(\mathbf{z}_{i,c} - \mu_{\mathbf{z},c})}{\sigma_{\mathbf{z},c}} + \beta_c,$ $\mu_{\mathbf{z}} = \frac{1}{B}\sum_{i=1}^{B} \mathbf{z}_i, \quad \sigma_{\mathbf{z}} = \sqrt{\frac{1}{k}\sum_{i=1}^{B}(\mathbf{z}_i - \mu_{\mathbf{z}})^2}.$	$\gamma, \beta \in \mathbb{R}^C.$

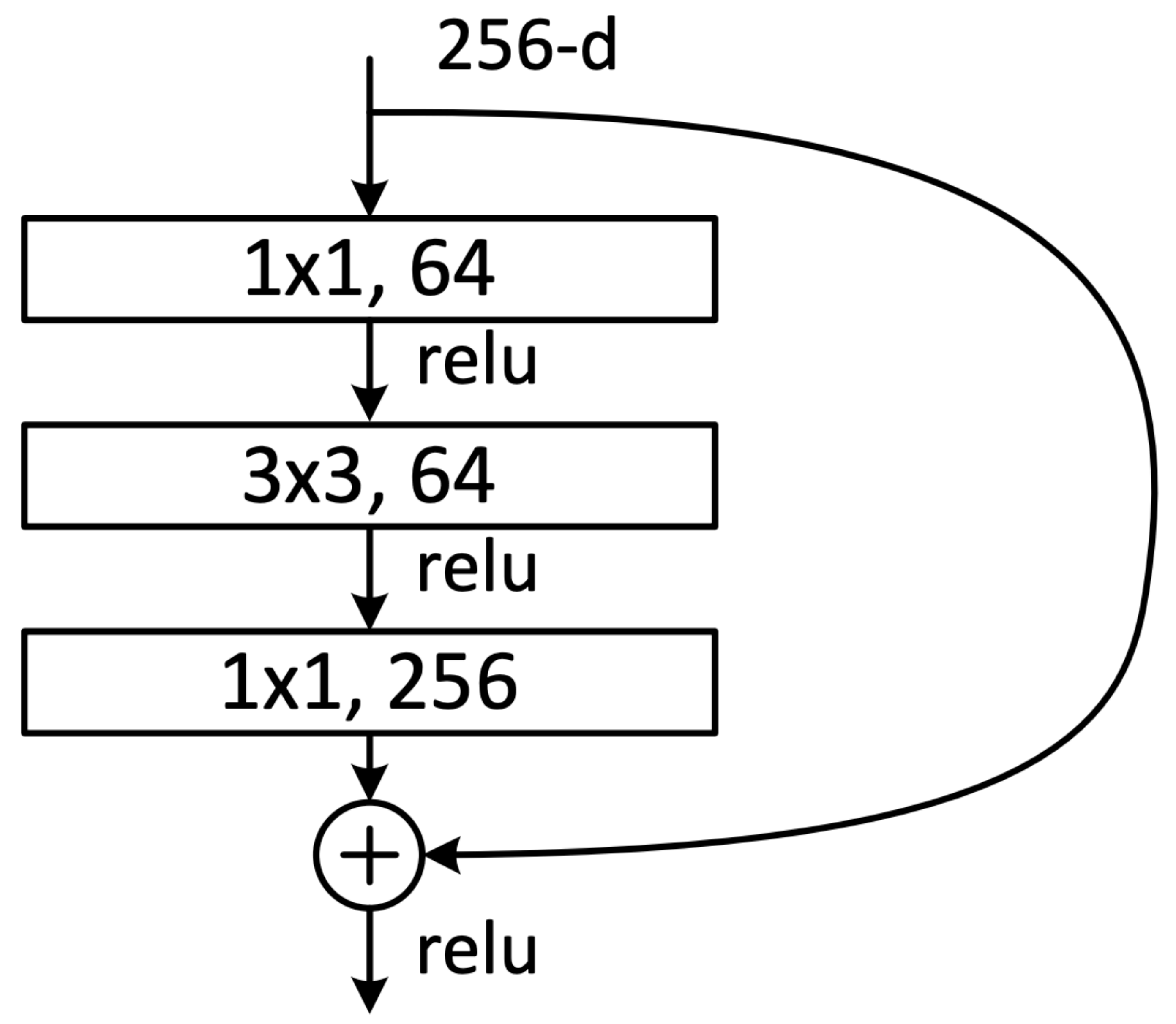
- Normalize activations of a batch to have learned mean β and variance γ^2.
- Applied per-batch. Compute mean, variance of full dataset after training.

A Modern Convnet Block

- If $\mathbf{x} \in \mathbb{R}^{C\times d\times d}$ then $f_\theta(\mathbf{x}) = \mathbf{z}$ where

Transformation	Weights
$\mathbf{u}'_{c,i,j} = \sum_{c'=1}^{C} \langle W^1_{c,c'}, \mathbf{x}_{i:i+1,j:j+1}\rangle,$	$W^1 \in \mathbb{R}^{C\times D\times 1\times 1},$
$\mathbf{u} = \mathrm{ReLU}\left(\mathrm{BatchNorm}(\mathbf{u}'; \gamma_1, \beta_1)\right),$	$\gamma_1, \beta_1 \in \mathbb{R}^D,$
$\mathbf{v}'_{c,i,j} = \sum_{c'=1}^{D} \langle W^2_{c,c'}, \mathrm{Pad}(\mathbf{u})_{i:i+k,j:j+k}\rangle,$	$W^2 \in \mathbb{R}^{D\times D\times k\times k},$
$\mathbf{v} = \mathrm{ReLU}\left(\mathrm{BatchNorm}(\mathbf{v}'; \gamma_1, \beta_1)\right),$	$\gamma_2, \beta_2 \in \mathbb{R}^D,$
$\mathbf{z}'_{c,i,j} = \sum_{c'=1}^{D} \langle W^3_{c,c'}, \mathrm{Pad}(\mathbf{v})_{i:i+1,j:j+1}\rangle,$	$W^3 \in \mathbb{R}^{D\times C\times 1\times 1}.$
$\mathbf{z} = \mathrm{ReLU}\left(\mathbf{x} + \mathrm{BatchNorm}(\mathbf{z}'); \gamma_3, \beta_3\right),$	$\gamma_2, \beta_2 \in \mathbb{R}^C.$

(C = 256, D = 64, k = 3)



He et. al., CVPR 2016

5-Minute Break

Convnet Hyper-Parameters

- Capacity of the network C.
- Inner convolutional capacity D.
- Convolution kernel size k.
- Depth of the convolution stack L.
- Size of a minibatch B (possible interaction with BatchNorm).
- All the usual hyper-parameters: dropout, l2, learning rates, initialization…

Parameterizing the VAE: Decoder

- Gaussian likelihood $p_\theta(x|z) = \mathcal{N}(x; g_\theta(z), \sigma_\theta^2(z)I)$.
- Let $f_{\text{in}} : \mathcal{Z} \to \mathbb{R}^{C\times d\times d}$ be a linear function.
- Let $h_\theta : \mathbb{R}^{C\times d\times d} \to \mathbb{R}^{C\times d\times d}$ be a convnet.
- Let $f_{\text{mean}} : \mathbb{R}^{C\times d\times d} \to \mathbb{R}^{C_{\text{out}}\times d\times d}$ be a 1x1 convolutional layer.
- Let $f_{\text{var}} : \mathbb{R}^{C\times d\times d} \to \mathbb{R}$ be a linear function (or just a constant).
- Define $g_\theta(z) = f_{\text{mean}} \circ h_\theta \circ f_{\text{in}}(z)$ and $\sigma_\theta(z) = f_{\text{var}} \circ h_\theta \circ f_{\text{in}}(z)$.

Parameterizing the VAE: Encoder

- Gaussian posterior approximation $q_\varphi(z|x) = \mathcal{N}(z; f_\varphi(x), \Sigma_\varphi(x))$.
- Let $f_{\text{in}} : \mathbb{R}^{C_{\text{in}} \times d \times d} \to \mathbb{R}^{C \times d \times d}$ be a 1x1 convolution layer.
- Let $h_\varphi : \mathbb{R}^{C \times d \times d} \to \mathbb{R}^{C \times d \times d}$ be a convnet.
- Let $f_{\text{mean}} : \mathbb{R}^{C \times d \times d} \to \mathcal{Z}$ be a linear function.
- Let $f_{\text{var}} : \mathbb{R}^{C \times d \times d} \to \mathcal{Z} \times \mathcal{Z}$ be a linear function.
- Define $f_\varphi(x) = f_{\text{mean}} \circ h_\varphi \circ f_{\text{in}}(x)$ and $\Sigma_\varphi^{1/2}(x) = f_{\text{var}} \circ h_\theta \circ f_{\text{in}}(x)$.

VAE Training

Algorithm 1: The Gaussian VAE.

while not converged **do**

Sample a minibatch $x_1, \ldots, x_B \sim p$, and $\varepsilon_1, \ldots, \varepsilon_B \sim \mathcal{N}(0, I_{\mathcal{Z}})$.

Compute (approximate) posterior samples $z_k = f_\varphi(x_k) + \Sigma_\varphi^{1/2}(x_k)\varepsilon_k$.

Update θ, φ via a gradient step on the following objective:

$$\sum_{k=1}^{B} \left[\frac{1}{2\sigma^2} \|x_k - g_\theta(z_k)\|^2 + D_{KL}(q_\varphi(z|x_k) \parallel r(z)) \right].$$

end

Sample $\tilde{z} \sim \mathcal{N}(0, I)$, $\eta \sim \mathcal{N}(0, I_{\mathcal{X}})$.

Output $\tilde{x} = g_\theta(\tilde{z}) + \sigma\eta$.