Conditioning Neural Networks with Feature-wise Affine Transformations

December 7, 2019

In this note, I summarize the Distill article "Feature-wise transformations" [4] and also some other papers that directly use the technique.

1 Conditioning

- Consider a general problem of computing a function that has two inputs, say $f(\mathbf{x}, \mathbf{y})$. We are interested in the problems of the form where $\mathbf{x}$ is the **main subject** of a problem; it is the thing to be analyzed or to make something out of. On the other hand, $\mathbf{y}$ is the **ocnditioning information**; it describes what to be done with the main subject. Examples of problems of this type include:
  - **Conditional GANs (cGANs).** Here, $\mathbf{x}$ is a latent code, and $\mathbf{y}$ is the class we want the generated image to be in. The output $f(\mathbf{x}, \mathbf{y})$ should be an image of class y.
  - **Conditional image translation.** x is an image, and y describes how we want the image to be transformed. For example, in the GANimation paper [11], $\mathbf{x}$ is an image of a human face, $\mathbf{y}$ is the Action Units (AUs) describing a facial expression, and $f(\mathbf{x}, \mathbf{y})$ should be the image of the same human with the expression changed to the one specified by $\mathbf{y}$.
  - **Style transfer**. $\mathbf{x}$ is the "content" image, $\mathbf{y}$ is the "style" image, and $f(\mathbf{x}, \mathbf{y})$ is an image with the content of $\mathbf{x}$ and the style of $\mathbf{y}$.
  - **Answering questions about images.** Again, $\mathbf{x}$ is an image. However, this time, $\mathbf{y}$ is a natural language question about the image such as "what material is the blue sphere made of?" [8].

  Performing computation with conditioning information is important, especially for the domain I'm interested in.
- One way to think about the conditioning information is that it describes the **tasks** to be done to the main subject.
- An easy way to include the conditioning information is to just concatenate $\mathbf{y}$ to $\mathbf{x}$ and feed that to the network that computes the function. This is easy for cGANs and conditinal image translation because it is easy to transform $\mathbf{y}$ to something that has the same format as $\mathbf{x}$. However, how would you do this for the question answering problem?
- There's a way to do the same thing as concatenating that is more general: **feature-wise transformation by adding bias**. After the input is processed by a number of layers, yielding immediate feature representation $g(\mathbf{x})$, we simply add a transformed value $\gamma(\mathbf{y})$ to $h(\mathbf{x})$ and then pass the transformed features to the next step:

$$f(\mathbf{x}, \mathbf{y}) := h(g(\mathbf{x}) + \gamma(\mathbf{y})).$$

Note that, when $\mathbf{x}$ and intermediate features are images, then $\gamma(\mathbf{y})$ is a vector with the same number of channels as $g(\mathbf{x})$, and the same $\gamma(\mathbf{y})$ is added to all spatial locations of $g(\mathbf{x})$. This is why we call the approach "feature-wise."

- Let's see why the above approach is the same as concatenating. Let us denote the concatenated input by

$$\tilde{\mathbf{x}} = \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}.$$

Invariably, the first layer of the network is a linear layer, which would compute:

$$W\tilde{\mathbf{x}} = \begin{bmatrix} W_1 & W_2 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = W_1\mathbf{x} + W_2\mathbf{y}.$$

Note that the same computation can be achieved by setting $\gamma(\mathbf{y}) := W_2\mathbf{y}$ and making the first linear layer having weight matrix W_1 instead of W.

- With the above equivalence, we can see that feature-wise transformation is more powerful than concatenation. This is because **it can be inserted to many places in the network**, allowing the conditioning information to be used in multiple steps. Concatenation, on the other hand, forces the network to preserve the conditioning information as it goes through the layers in order to do so.

- We can generalize feature-wise transformation by adding bias by adding a scalign factor to get **feature-wise transformation by affine transformation** or ***f*eature-w*i*se *l*inear *m*odulation (FiLM)** as called by Perez et al. [10] as follows:

$$f(x, y) = h\big(\beta(\mathbf{y}) \odot g(\mathbf{x}) + \gamma(\mathbf{y})\big)$$

where $\odot$ denotes the element-wise multiplication. When dealing with images, $\odot$ multiplies the same $\beta(\mathbf{y})$ to all spetial locations of $g(\mathbf{x})$.

- FiLM is a general framework that subsumes many previous similar techniques such as conditional batch normalization (CBN) and adaptive instance normalization (AdaIN). In the following sections, we will summarize papers those papers.

2 Conditional Batch Normalization

- *Conditional batch normalization* (CBN) is used heavily by de Vries et al. [3]. The problem they tackled is *visual question answering* (VQA) in which the system is given an image and a natural language question about the image. The goal is for the system to answer the question. (We'll make this more specific later.)

- One common way to tackle VQA is to compute two feature vectors: one from the image, and the other from the question. Then, these two features would be fused, and the answer would be derived from the fused feature vector. This approach means that the question has no impact on the pipeline that processes the image, and the content of the question is only incorporated at the very large stage of the pipeline.

- However, there are psychological evidences that natural language cues (e.g., hearing some words) can influence how human process visual signals from the eyes down to low-level features. This suggests that it is advisable to have the question influence how a network processes the image an early stage.

- The authors worked on two datasets:
  - VQA v1 [1]

    - 614K question on 204K images.
    - There are three types of questions:
        - Yes/no: Is this a vegetarian pizza?
        - Numerical answer: How many slices of pizza are there?
        - Free form: What is just under the tree?
    - Here, the length of the answers are at most 3 words.
    - From the official implementation of the paper, it seems that the last layer of the network is a softmap layer, so it might not be able to answer free from questions with multiple words answer.
    - The dataset contains common sense questions that can be answered without the image.
  - GuessWhat?! [2]
    - 822K yes/no questions on 67K images.
    - This is based on two player game. There are two players: the questioner and the Oracle. Both players see the same image. The Oracle is assigned an object in the scene. The questioner is to find what the object is by asking the Oracle yes/no questions. The Oracle can answer yes, no, or not applicable.
    - Example questions:
        - Is it an elephant?
        - Is in on the left?
        - Can you see the person's face?
        - Is she holding it?
        - The log on the right?
        - A chair?
    - The dataset contains fewer common sense questions than VQA v1.

2.1 The Algorithm

- The input question is process to get an embedding.
  - A question $\mathbf{q} = [w_k]_{k=1}^K$ is a sequence of tokens w_k from a predefined vocabulary V.
  - Each token is transformed into a dense word embedding $e(w_k)$. The details of this is not that important (at least for me).
  - The sequence of dense embedding $[e(w_k)]_{k=1}^K$ is then fed to a RNN. The RNN used in the paper is the LSTM.
  - The last hidden state of the LSTM is taken as the embedding of the question. Let us denote this by $\mathbf{e_q}$.
- For the GuessWhat?! dataset, the image given to the nnetwork is the original image cropped around the object the Oracle is assigned. To make the network able to answer questions as an Oracle, the paper computes an embedding of the crop window and the class of the object. Then, it concatenates that vector to $\mathbf{e_q}$ as computed above to produce the final embedding of the question.
- The paper uses a pretrained ResNet (ResNet-50) to convert the image to answer.
- ResNet employs *batch normalization*.
  - The input is a feature map $F \in \mathbb{R}^{N \times C \times H \times W}$. Let is denote an element of this feature map‘ by $F_{n,c,h,w}$.

– Batch normalization is given by the following function:

$$BN(F_{n,c,h,w}|\gamma_c, \beta_c) := \gamma_c \frac{F_{n,c,w,h} - E_{n,h,w}[F_{\cdot,c,\cdot,\cdot}]}{\sqrt{\text{Var}_{n,h,w}(F_{\cdot,c,\cdot,\cdot}) + \epsilon}} + \beta_c$$

where β_c and γ_c are learned parameters. At test time, where the batch may not be available, the batch mean and variance are replaced by the exponentially moving average computed during training.

- The pretrained ResNet is connected to the question embedding through **conditional batch normalization**: the batch normalization parameters of all batch normalization units are changed by the following procedure.
  – The embedding $\mathbf{e_q}$ is fed to two one-hidden-layer multi-layer perceptron to produce changes to β_c and γ_c:

$$\Delta\beta_c = MLP(\mathbf{e_q})$$
$$\Delta\gamma_c = MLP(\mathbf{e_q}).$$

  – Then, batch normalization is computed by $BN(F_{n,c,h,w}|\hat{\beta}_c, \hat{\gamma}_c)$ where:

$$\hat{\beta}_c = \beta_c + \Delta\beta_c$$
$$\hat{\gamma}_c = \gamma_c + \Delta\gamma_c$$

  Parameters for the MLPs only consist of 1% of the total parameters of the ResNet.

- Besides the changed batch normalization parameters, all the other parameters of the ResNet are fixed.
- After the input image is processed by the (CBNed) ResNet-50, the paper extracts the last feature maps F of the layer before the last pooling layer. The feature map is of size 7×7.
- An attention map is then computed from F conditioned on the embedding $\mathbf{e_q}$ of the question:

$$\xi_{h,w} = MLP(\text{concat}(\mathbf{F}_{i,\cdot,h,w}, \mathbf{e_q})),$$
$$\alpha_{h,w} = \frac{\exp(\xi_{h,w})}{\sum_{h,w} \exp(\xi_{h,w})}.$$

  For the MLP function, the paper uses a multi-layer perceptron with one hidden layer.

- The paper then uses the attention map to contract the feature map into one vector:

$$\mathbf{e_v} = \sum_{h,w} \alpha_{h,w} \mathbf{F}_{i,\cdot,h,w}.$$

- The two embedding vectors are then fused:

$$\text{fuse}(\mathbf{e_q}, \mathbf{e_v}) = P\big(\tanh(U\mathbf{e_q}) \odot \tanh(V\mathbf{e_q})\big) + \mathbf{b}_P$$

  where P, U, V are trainable weight matrices, and $\mathbf{b}_P$ are trainable bias vector.

- The fused vector is then passed to a linear layer and then a softmax to produce a probability distribution over the discrete answers.

- The authors also used another algorithm [9] (the paper calls it MRN) to produce $\mathbf{e_v}$: instead of looking at the feature maps once, the network would look at them in glimpses:

$$\begin{aligned}\xi^g_{h,w} &= P_g\big(\tanh(U'\mathbf{e_q}) \odot \tanh(V'\mathbf{F}_{i,\cdot,h,w})\big)\\ a^g_{h,w} &= \frac{\exp(\xi^g_{h,w})}{\sum_{h,w}\exp(\xi^g_{h,w})}\\ \mathbf{e_v} &= \Big\|_g \sum_{h,w} \alpha^g_{h,w}\mathbf{F}_{i,\cdot,h,w}.\end{aligned}$$

  where P_g is a trainable weight matrix for glimpse g, U' and V' are trainable weight matrices shared among the glimpsed, and the operator $\|$ denotes vector concatenation.

2.2 Results

- Using CBN with ResNet-50 achieved about 2% accuracy improvement over other SOTA methods on the VQA v1 dataset.
- For GuessWhat?!, the errors dropped also by about 3% from that of the network that does not perform CBN at all.
- When t-SNE is performed on the feature maps before the attention mechanism (i.e., $\alpha_{h,w}$), we can observe that they form clusters based on answer types in ResNet with CBN. However, the same is not true for raw ResNet.
- Performance slowly decreases when CBN is applied exclusively to later stages.

3 Conditional Instance Normalization

- Ghiasi et al. [6] is a paper using *conditional instance normalization* (CIN) in order to perform image stylization.
- The inputs are two images. The *content image* is typically a photograph, and the *style image* is typically a painting. We would like to compute an output image containing the scene and objects of the content image but looks like it was painted in the style of the style image.
- Let c denote the content image, s the style image, and x the output image. The loss function for the style transfer task is given by:

$$\min_x \mathcal{L}_c(x,c) + \lambda_s \mathcal{L}_s(x,s)$$

  where $\mathcal{L}_c$ denotes the content loss, and $\mathcal{L}_s$ denotes the style loss. λ_s is a hyperparameter specifying the relative importance of the style in the output image. The two sublosses are given by:

$$\begin{aligned}\mathcal{L}_c(x,c) &= \sum_{j\in\mathcal{C}} \frac{1}{n_j}\|f_j(x) - f_j(x)|_2^2\\ \mathcal{L}_s(x,s) &= \sum_{i\in\mathcal{S}} \frac{1}{n_i}\|\mathcal{G}[f_i(x)] - \mathcal{G}[f_i(s)]\|_F^2.\end{aligned}$$

  Here, $\mathcal{C}$ is a set of higher layers in an image classification network, and $\mathcal{S}$ is the set of lower layers. (The paper claims that contents are represented by higher layers and styles are represented by lower layers.) f_l denotes the feature map that the output of Layer l of the classification network, and n_l denotes the number of elements in the feature map. Lastly, $\mathcal{G}[f_l(x)]$ is the Gram matrix associated with $f_l(x)$.

- The style transfer network used in the paper has an encoder-decoder architecture. However, instead of using the standard batch normalization units, it uses a conditional instance normalization where the unit's output is given by:

$$\tilde{z} = \gamma_s \frac{z - \mu}{\sigma} + \beta_s$$

where μ and σ are the per-channel mean and standard deviation across the pixels of the input feature map z. The parameters γ_s and β_s define an affine transformation on the normalized feature map, specialized to a specific style image.

- The collection of β_s and γ_s for all the normalization layers forms a 3000-dimensional vector, forming a space of artistic styles that can be interpolated. Dumoulin et al. [5] learned such vectors for 32 paintings, allowing them to use the network to transfer photos to the styles of these 32 paintings and also the interpolated styles.

- Ghiasi et al. extends the work of Dumoulin et al. by proposing a style transfer network $P(\cdot)$ which takes a style image s and compute the normalization parameters.

  In particular, they feed s to a pretrained Inception-v3 network and compute the mean across each channel of the Mixed-6e layer, resulting in a vector with dimension 768. This is then passed to two fully connected layers to compute the embedding parameters. The first fully connected layer has only 100 units in order to compress the representation.

- In summary, the results of the paper are as follows:
  - The authors were able to perform style transfer on arbirary content and style images. This was not possible before 2017.
  - The network generalizes to styles of unobserved paintings.
  - Training with more style images does not improve losses. However, it does improve generalization to unseen style images. The losses on unobserved paintings plateaus after about 16,000 style images are used.
  - The space of normalization parametersr capture semantic structure of styles and allow for exploration.

4 Adaptive Instance Normalization

- *Adaptive instance normalization* (AdaIN) is proposed by Huang and Belongie [7]. It is very similar to CIN.

- The AdaIN unit takes in two inputs:
  - x is the feature map (computed somehow) of the content image c.
  - y is the feature map (computed in the same way as the content image) of the style image s.

  The unit matches the per-channel means and variances of x to those of y:

$$\text{AdaIN}(x, y) = \sigma(y) \frac{x - \mu(x)}{\sigma(x)} + \mu(y)$$

  where $\mu(\cdot)$ and $\sigma(\cdot)$ denote the per-channel mean and standard deviation of the input.

- The style transfer network of Huang and Belongie also has a encoder-decoder architecture.
  - The encoder network f is the first few layers of pretrained VGG-19. The paper uses up to **relu4_1**. This part is fixed.

  - The first step of processing is to compute $f(c)$ and $f(s)$.
  - AdaIN is then used to match the statistics of $f(c)$ to that of $f(s)$:

$$t = \text{AdaIN}(f(c), f(s)).$$

  - Then, t is passed to a decoder g to get the final image. g is the only part of the network that needs to be trained.

- The loss function used to train g is given by:

$$\mathcal{L} = \mathcal{L}_c + \lambda \mathcal{L}_s.$$

  The first term is the content loss. It encourages the decoder image to preserve the content with respect to the encoder:

$$\mathcal{L}_c = \|f(g(t)) - t\|_2.$$

  The second term is the style loss. It encourages the decoded image to match the statistics of the style image, after being processed by several layers of VGG-19:

$$\mathcal{L}_s = \sum_{i=1}^{L} \|\mu(\phi_i(g(t))) - \mu(\phi_i(s))\|_2 + \sum_{i=1}^{L} \|\sigma(\phi_i(g(t))) - \sigma(\phi_i(s))\|_2$$

  where ϕ_i denotes output of a layer in VGG-19. The paper uses `relu1_1`, `relu2_1`, `relu3_1`, and `relu4_1`. The interesting thing is that it doesn't use any Gram matrices at all.

- Compared to CIN in Ghiasi et al.'s paper, AdaIN is simpler but is less flexible. Ghiasi et al.'s paper observes that CIN is better at reducing both content and style losses than AdaIN.

5 Feature-wise Linear Modulation

- *Feature-wise Linear Modulation* (FiLM, centainly a confusing acronym) is introduced by Perez et al. [10]. It is a genalization of CIN that basically decouples the affine transformation from normalization.

- The task the paper tackles is VQA. However, they work on the CLEVR dataset [8]. The dataset contains images of rendered objects with simple shapes and textures, but the questions are complex and require logic to answer. Example questions include:

  - What number of cylinders are small purple things or yellow rubber things?
  - What color is the other object that is the same shape as the large brown matte thing?

- FiLM proposes modifying the intermediate feature map in the image processing network by per-channel affine transformation:

$$\text{FiLM}(\mathbf{F}_{i,c,\cdot,\cdot}|\gamma_{i,c}, \beta_{i,c}) = \gamma_{i,c}\mathbf{F}_{i,c,\cdot,\cdot} + \beta_{i,c}$$

  Here, $\gamma_{i,c}$ and $\beta_{i,c}$ are scalars computed from feeding two learned functions of another input $\mathbf{x}_i$ (that is, the question):

$$\gamma_{i,c} = f_c(\mathbf{x}_i)$$
$$\beta_{i,c} = h_c(\mathbf{x}_i).$$

- In the paper, f and h are neural networks. First, the question is fed to a GRU, token by token, and the 4096-dimensional internal state of the GRU is extracted. This vector is then passed to leanred affine transformation to produce $\gamma^n_{i,c}$ and $\beta^n_{i,c}$ for the nth modification points.

- The image processing network takes in a 224×224 image, and it extracts a $128 \times 14 \times 14$ feature map. The network that does the extraction is either:
  - A network trained from scratch, containing 4 blocks, each with a 4×4 convolutional unit that halves the feature map size each time followed by a ReLU and a batch normalization.
  - ResNet-101, pretrained on ImageNet, up to *conv4*.
- The feature map from the last step is then processed that 4 residual blocks with FiLM units. The output is then passed to:
  - A 1×1 convolution to increase the number of channels to 512. (Size $= 512 \times 14 \times 14$.)
  - A global max pooling unit. (Size $= 512 \times 1 \times 1$.)
  - Two-layer MLP with 1024 hidden units.
  - Softmax to output distribution over the answers.
- The residual block is used in the above step has the following train of units:
  - 1×1 convolution.
  - ReLU.
  - 3×3 convolution.
  - Batch normalization (without affine transformation).
  - FiLM
  - ReLU.

  As usual, the output of this train of units are added back to the original input.
- Their FiLMed network outperformed humans and halved SOTA errors.

References

[1] Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C. L., and Parikh, D. VQA: visual question answering. *CoRR abs/1505.00468* (2015).

[2] de Vries, H., Strub, F., Chandar, S., Pietquin, O., Larochelle, H., and Courville, A. C. Guesswhat?! visual object discovery through multi-modal dialogue. *CoRR abs/1611.08481* (2016).

[3] de Vries, H., Strub, F., Mary, J., Larochelle, H., Pietquin, O., and Courville, A. C. Modulating early visual processing by language. *CoRR abs/1707.00683* (2017).

[4] Dumoulin, V., Perez, E., Schucher, N., Strub, F., Vries, H. d., Courville, A., and Bengio, Y. Feature-wise transformations. *Distill* (2018). https://distill.pub/2018/feature-wise-transformations.

[5] Dumoulin, V., Shlens, J., and Kudlur, M. A learned representation for artistic style. *CoRR abs/1610.07629* (2016).

[6] Ghiasi, G., Lee, H., Kudlur, M., Dumoulin, V., and Shlens, J. Exploring the structure of a real-time, arbitrary neural artistic stylization network. *CoRR abs/1705.06830* (2017).

[7] Huang, X., and Belongie, S. J. Arbitrary style transfer in real-time with adaptive instance normalization. *CoRR abs/1703.06868* (2017).

[8] Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Zitnick, C. L., and Girshick, R. B. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. *CoRR abs/1612.06890* (2016).

[9] Kim, J., On, K. W., Lim, W., Kim, J., Ha, J., and Zhang, B. Hadamard product for low-rank bilinear pooling. *CoRR abs/1610.04325* (2016).

[10] Perez, E., Strub, F., de Vries, H., Dumoulin, V., and Courville, A. C. Film: Visual reasoning with a general conditioning layer. *CoRR abs/1709.07871* (2017).

[11] Pumarola, A., Agudo, A., Martinez, A., Sanfeliu, A., and Moreno-Noguer, F. Ganimation: One-shot anatomically consistent facial animation.