Interpretability, VAEs, and GANs

Recitation 3

MIT - 6.802 / 6.874 / 20.390 / 20.490 / HST.506 - Spring 2021

Dylan Cable

Variational Inference

Slide credit to – Jeff Miller, Harvard Biostatistics

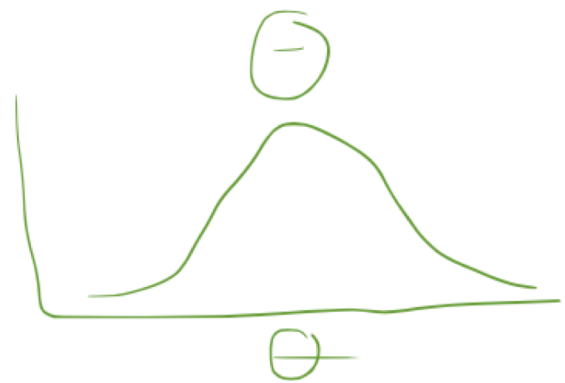
$p(\theta|x)$

Posterior

Params Data

$p(\theta|x) \propto p(\theta)p(x|\theta)$

$p(x|\theta)$ Likelihood



Posterior

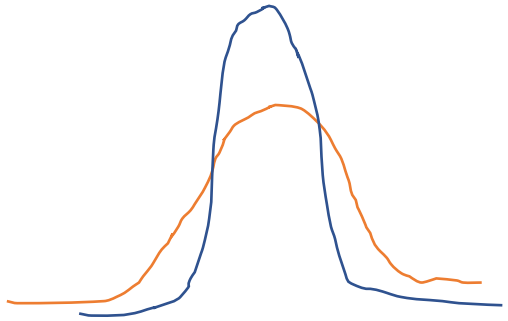
Introduction

- Variational inference (VI) is an approach to posterior inference based on approximating the posterior by a nicer distribution.

- The general idea is to:
  1. choose a nice family of distributions $\mathcal{Q}$,
  2. find a $q \in \mathcal{Q}$ that is as close as possible to the posterior, and
  3. use q to quantify uncertainty, as a proxy for the posterior.

- Different choices of $\mathcal{Q}$ and definitions of "close" lead to different variational inference techniques.

- In Bayesian statistics, VI is also known as variational Bayes, but VI is also useful outside of Bayesian inference.



$$\log\left(\frac{q_f(x)}{\pi(x)}\right) \geq T$$

optimal test

Likelihood Ratio Test

$$D(q_0 \| \pi) = \int q_0(x) \log\left(\frac{q_0}{\pi}\right)$$

Classic variational inference

- Let $\pi(\theta)$ denote the target distribution, e.g., $\pi(\theta) = p(\theta|x)$.

- Classic VI makes the following choices:
  1. the approximating family $\mathcal{Q}$ consists of all factorized distributions of the form

     $$q(\theta) = q_1(\theta_1) \cdots q_m(\theta_m)$$

     for some decomposition of θ into components $\theta_1, \ldots, \theta_m$, and

  2. we seek the $q \in \mathcal{Q}$ that minimizes the Kullback–Leibler divergence from q to π,

     $$q^{\text{opt}} \in \operatorname*{argmin}_{q \in \mathcal{Q}} D(q\|\pi).$$

- The *Kullback–Leibler (KL) divergence*, or *relative entropy*, is

$$D(q\|\pi) = \int q(\theta) \log \frac{q(\theta)}{\pi(\theta)} d\theta = \mathrm{E}_q\Big(\log \frac{q(\theta)}{\pi(\theta)} \Big).$$

Toy example: VI for univariate normal

- Consider the univariate normal model as a toy example:

$$X_1, \ldots, X_n | \mu, \lambda \overset{\text{iid}}{\sim} \mathcal{N}(\mu, \lambda^{-1})$$

and assume an improper uniform prior on μ and λ.

- Define the target distribution to be the posterior:

$$\begin{aligned} \pi(\mu, \lambda) = p(\mu, \lambda | x_{1:n}) &\propto p(x_{1:n} | \mu, \lambda) \\ &\underset{\mu,\lambda}{\propto} \lambda^{n/2} \exp\big(-\tfrac{1}{2}\lambda \textstyle\sum_{i=1}^n (x_i - \mu)^2 \big). \end{aligned}$$

- Thus, $\log \pi(\mu, \lambda) = \frac{n}{2} \log \lambda - \frac{1}{2}\lambda \sum_{i=1}^n (x_i - \mu)^2 + \text{const.}$

- A natural decomposition to try would be $\theta_1 = \mu$ and $\theta_2 = \lambda$, that is, to consider approximations of the form

$$q(\mu, \lambda) = q_1(\mu) q_2(\lambda).$$

For convenience, we write $q(\mu, \lambda) = q(\mu) q(\lambda)$.

Classic variational inference: Algorithm

- To derive the classic VI updates, we need only specify the target distribution π and the decomposition of θ into $\theta_1, \ldots, \theta_m$.

- The classic VI algorithm then proceeds as follows:
  1. Initialize $q_1, \ldots, q_m$.
  2. Repeat until convergence:
     For $j = 1, \ldots, m$, update q_j by setting it to be

$$q_j^{\text{new}}(\theta_j) \propto \exp(h_j(\theta_j))$$

     where

$$h_j(\theta_j) := \mathrm{E}_q\big(\log \pi(\theta) \mid \theta_j\big) = \int \big(\log \pi(\theta)\big) \prod_{i \neq j} q_i(\theta_i) d\theta_i.$$

  3. Use $q(\theta) = q_1(\theta_1) \cdots q_m(\theta_m)$ as an approximation to $\pi(\theta)$.

Justification of classic VI algorithm (1/2)

- Let $\theta = (\theta_1, \ldots, \theta_m)$ where $\theta_i \sim q_i$ independently for some $q_1, \ldots, q_m$. Let π be the target distribution. Then for any j,

$$
\begin{aligned}
D(q_1 \cdots q_m \,\|\, \pi) &= \mathrm{E}\Big(\log \frac{q_1(\theta_1) \cdots q_m(\theta_m)}{\pi(\theta)} \Big) \\
&= \sum_{i=1}^{m} \mathrm{E} \log q_i(\theta_i) - \mathrm{E} \log \pi(\theta) \\
&= \mathrm{E} \log q_j(\theta_j) - \mathrm{E}\big(\mathrm{E}(\log \pi(\theta) | \theta_j)\big) + (\text{const wrt } q_j) \\
&= \mathrm{E} \log q_j(\theta_j) - \mathrm{E} h_j(\theta_j) + (\text{const wrt } q_j) \\
&= \mathrm{E}\Big(\log \frac{q_j(\theta_j)}{c\, e^{h_j(\theta_j)}} \Big) + (\text{const wrt } q_j) \\
&= D(q_j \,\|\, c\, e^{h_j}) + (\text{const wrt } q_j)
\end{aligned}
$$

where $h_j(\theta_j) := \mathrm{E}(\log \pi(\theta) \mid \theta_j)$ and $1/c = \int e^{h_j(\theta_j)} d\theta_j$.

Justification of classic VI algorithm (2/2)

- Hence,

$$\operatorname*{argmin}_{q_j} D(q_1 \cdots q_m \,\|\, \pi) = \operatorname*{argmin}_{q_j} D(q_j \,\|\, c\, e^{h_j}).$$

- By the properties of KL divergence, $D(q_j \,\|\, c\, e^{h_j}) \geq 0$ with equality if and only if $q_j = c\, e^{h_j}$ almost everywhere.

- Therefore, choosing $q_j \propto e^{h_j}$ minimizes $D(q_1 \cdots q_m \,\|\, \pi)$ with respect to q_j, given q_i for $i \neq j$.

- This shows that the classic VI algorithm performs coordinate descent, updating q_j to minimize KL at each step, holding q_i fixed for $i \neq j$.

Classic variational inference: Algorithm

- To derive the classic VI updates, we need only specify the target distribution π and the decomposition of θ into $\theta_1, \ldots, \theta_m$.

- The classic VI algorithm then proceeds as follows:
  1. Initialize $q_1, \ldots, q_m$.
  2. Repeat until convergence:
     For $j = 1, \ldots, m$, update q_j by setting it to be

$$q_j^{\text{new}}(\theta_j) \propto \exp(h_j(\theta_j))$$

     where

$$h_j(\theta_j) := \mathrm{E}_q\big(\log \pi(\theta) \mid \theta_j\big) = \int \big(\log \pi(\theta)\big) \prod_{i \neq j} q_i(\theta_i) d\theta_i.$$

  3. Use $q(\theta) = q_1(\theta_1) \cdots q_m(\theta_m)$ as an approximation to $\pi(\theta)$.

Toy example: VI for univariate normal

- Consider the univariate normal model as a toy example:

$$X_1, \ldots, X_n | \mu, \lambda \overset{\text{iid}}{\sim} \mathcal{N}(\mu, \lambda^{-1})$$

and assume an improper uniform prior on μ and λ.

- Define the target distribution to be the posterior:

$$\begin{aligned} \pi(\mu, \lambda) = p(\mu, \lambda | x_{1:n}) &\propto p(x_{1:n} | \mu, \lambda) \\ &\underset{\mu,\lambda}{\propto} \lambda^{n/2} \exp\big(-\tfrac{1}{2}\lambda \textstyle\sum_{i=1}^n (x_i - \mu)^2 \big). \end{aligned}$$

- Thus, $\log \pi(\mu, \lambda) = \frac{n}{2} \log \lambda - \frac{1}{2}\lambda \sum_{i=1}^n (x_i - \mu)^2 + \text{const.}$

- A natural decomposition to try would be $\theta_1 = \mu$ and $\theta_2 = \lambda$, that is, to consider approximations of the form

$$q(\mu, \lambda) = q_1(\mu) q_2(\lambda).$$

For convenience, we write $q(\mu, \lambda) = q(\mu) q(\lambda)$.

Normal example: Deriving the VI updates (1/3)

- Updating $q(\mu)$ given $q(\lambda)$:

$$\begin{aligned} h_1(\mu) &= \int q(\lambda) \log \pi(\mu, \lambda) d\lambda \\ &= \frac{n}{2} \int q(\lambda) \log \lambda - \frac{1}{2} \left(\sum_i (x_i - \mu)^2 \right) \int \lambda q(\lambda) d\lambda + \text{const.} \end{aligned}$$

- Therefore, according to the algorithm, we update $q(\mu)$ to be

$$\begin{aligned} q^{\text{new}}(\mu) &\propto \exp(h_1(\mu)) \propto \exp\left(-\tfrac{1}{2} \mathrm{E}(\lambda) \textstyle\sum_i (x_i - \mu)^2 \right) \\ &\underset{\mu}{\propto} \mathcal{N}\left(\mu \mid \bar{x}, \, (n\mathrm{E}(\lambda))^{-1}\right). \end{aligned}$$

- Computationally, we only need to compute and store $\bar{x}$ and $\mathrm{E}(\lambda)$.

- Here, $\mathrm{E}(\lambda) = \int \lambda q(\lambda) d\lambda$ is computed using the current $q(\lambda)$.

Normal example: Deriving the VI updates (2/3)

- Updating $q(\lambda)$ given $q(\mu)$:

$$\begin{aligned} h_2(\lambda) &= \int q(\mu) \log \pi(\mu, \lambda) d\mu \\ &= \frac{n}{2} \log \lambda - \frac{1}{2} \lambda \sum_i \int (x_i - \mu)^2 q(\mu) d\mu + \text{const.} \end{aligned}$$

- Therefore, according to the algorithm, we update $q(\lambda)$ to be

$$\begin{aligned} q^{\text{new}}(\lambda) &\propto \exp(h_2(\lambda)) \propto \lambda^{n/2} \exp(-\tfrac{1}{2} S \lambda) \\ &\underset{\lambda}{\propto} \text{Gamma}(\lambda \mid n/2 + 1,\ S/2) \end{aligned}$$

  where, after plugging in $q(\mu) = \mathcal{N}(\mu \mid \bar{x},\ (n\text{E}(\lambda))^{-1})$ and simplifying,

$$S = \sum_i \int (x_i - \mu)^2 q(\mu) d\mu = n\hat{\sigma}^2 + 1/\text{E}(\lambda).$$

- Here, $\hat{\sigma}^2 = \frac{1}{n} \sum_i (x_i - \bar{x})^2$.

Normal example: Deriving the VI updates (3/3)

- Thus, the updates to $q(\mu)$ and $q(\lambda)$ are:

$$q^{\text{new}}(\mu) = \mathcal{N}\big(\mu \mid \bar{x},\, (n\mathrm{E}(\lambda))^{-1}\big),$$
$$q^{\text{new}}(\lambda) = \text{Gamma}\left(\lambda \mid n/2+1,\ \tfrac{1}{2}(n\hat{\sigma}^2 + 1/\mathrm{E}(\lambda))\right).$$

- The only thing we need to compute at each iteration is $\mathrm{E}(\lambda)$.

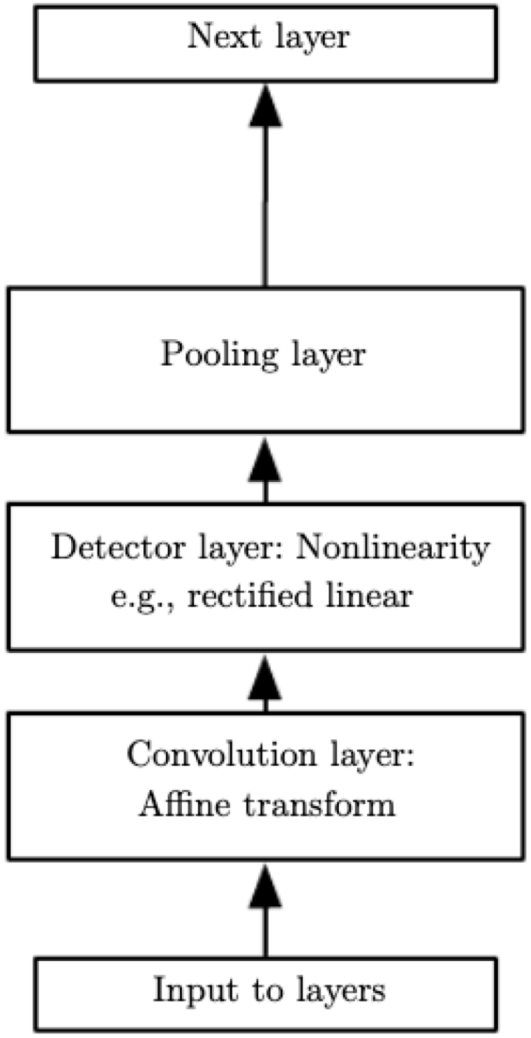
- From the form of $q^{\text{new}}(\lambda)$, we see that

$$\mathrm{E}^{\text{new}}(\lambda) = \frac{n/2+1}{\frac{1}{2}(n\hat{\sigma}^2 + 1/\mathrm{E}(\lambda))}.$$

- In this example, it turns out that we can analytically solve for the limiting value of $\mathrm{E}(\lambda)$, which is $\mathrm{E}(\lambda) = (n+1)/(n\hat{\sigma}^2)$.

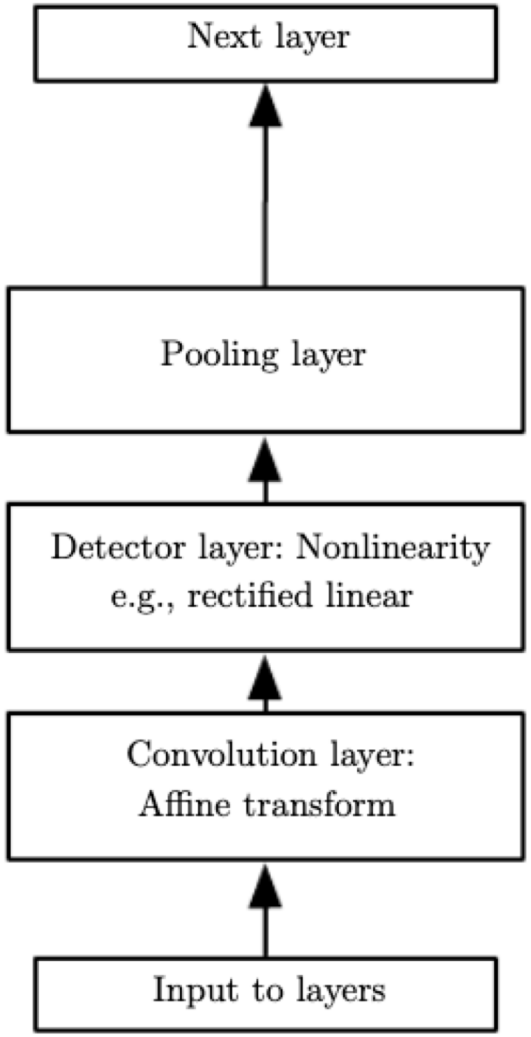
- However, in more complex models, we will have to iterate until convergence.

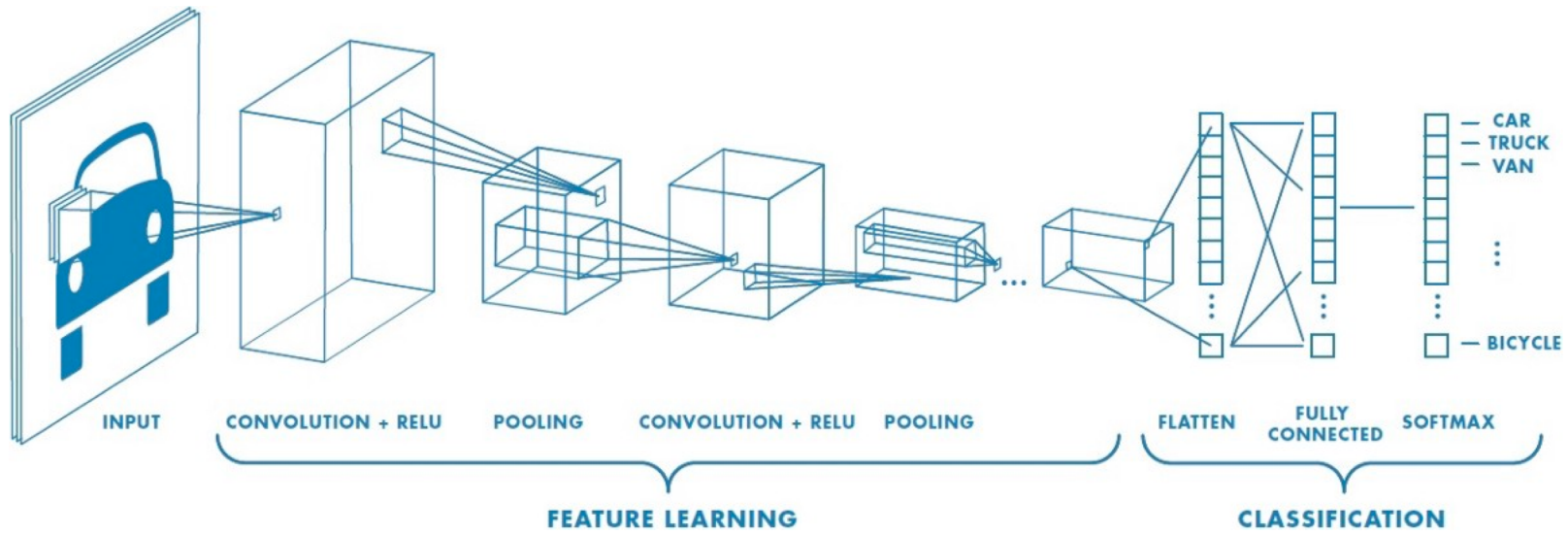
Convolutional neural networks: overview



https://www.deeplearningbook.org/contents/convnets.html

Convolutional neural networks: overview





https://www.mathworks.com/solutions/deep-learning/convolutional-neural- network.html

Convolutional neural networks: convolution operation

Imagine a position x taken at a time point t, which are both continuous,

$$x(t)$$

Suppose this measurement is noisy. To obtain a less noisy estimate of the position, we want a weighted average of recent measurements. Using a weighting function $w(a)$, where *a* is the age of the measurement,

$$s(t) = \int x(a)w(t-a)da = (x * w)(t)$$

input

convolutional kernel/filter

Generally, sensor inputs are not continuous and most signal processing tasks rely on discretized data—data provided at regular intervals. In machine learning, input is generally a multi-dimensional array of data and the kernel is a multi-dimensional array of parameters:

$$S(i,j) = (W * I)(i,j) = \sum_m \sum_n I(i+m, j+n)W(m,n)$$

This operation also exhibits translational equivariance. Let *g* be a function mapping one image function to another image function that shifts inputs to the right by one,

$$I'(i,j) = g(I(i,j)) = I(i-1,j)$$

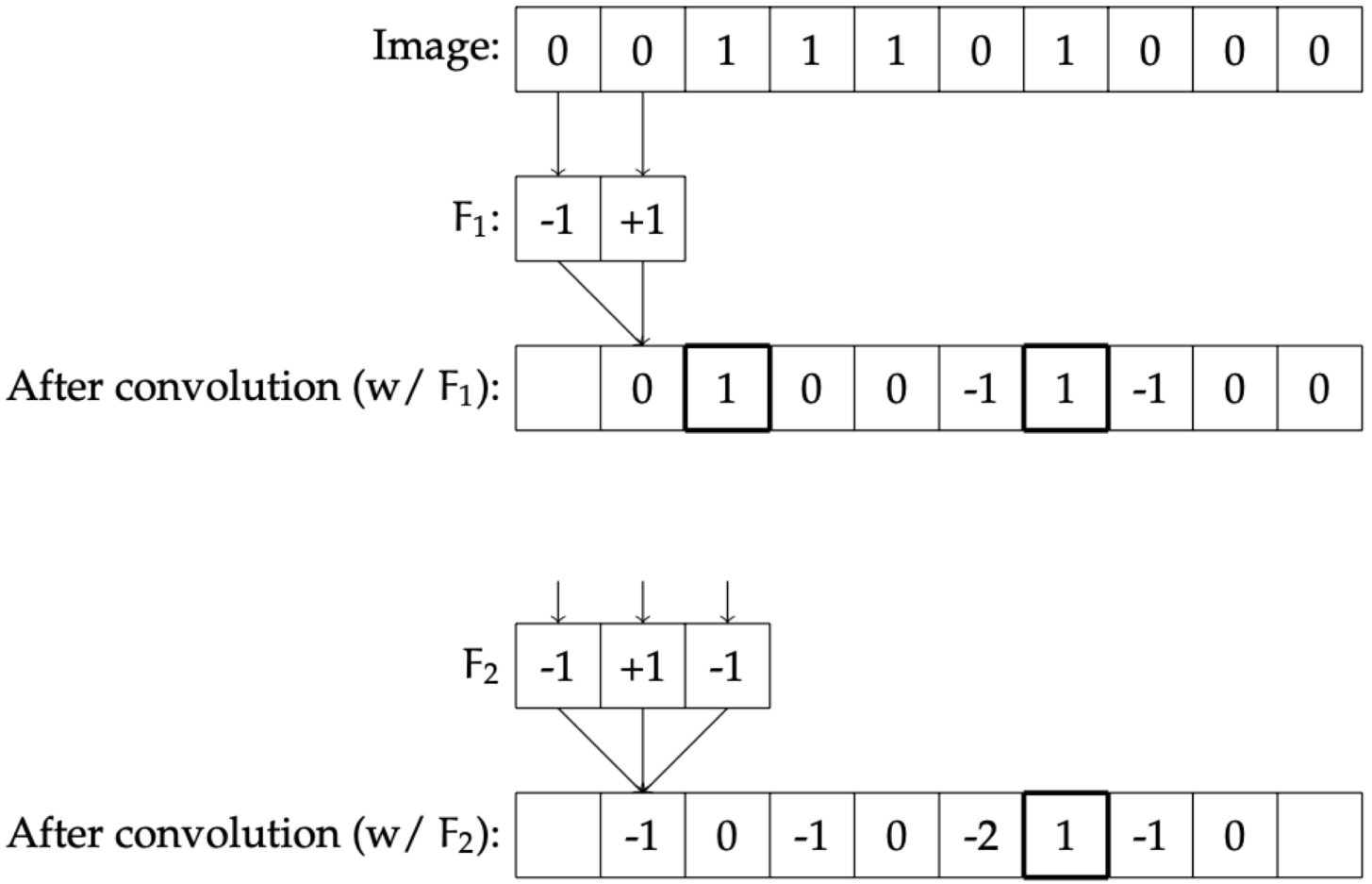
If we apply the transformation to *I*, then apply a convolution, the output is equivalent to applying a convolution to *I'* then applied the transformation *g* to the output,

$$\left(W * I'\right)(i,j) = g((W * I)(i,j))$$

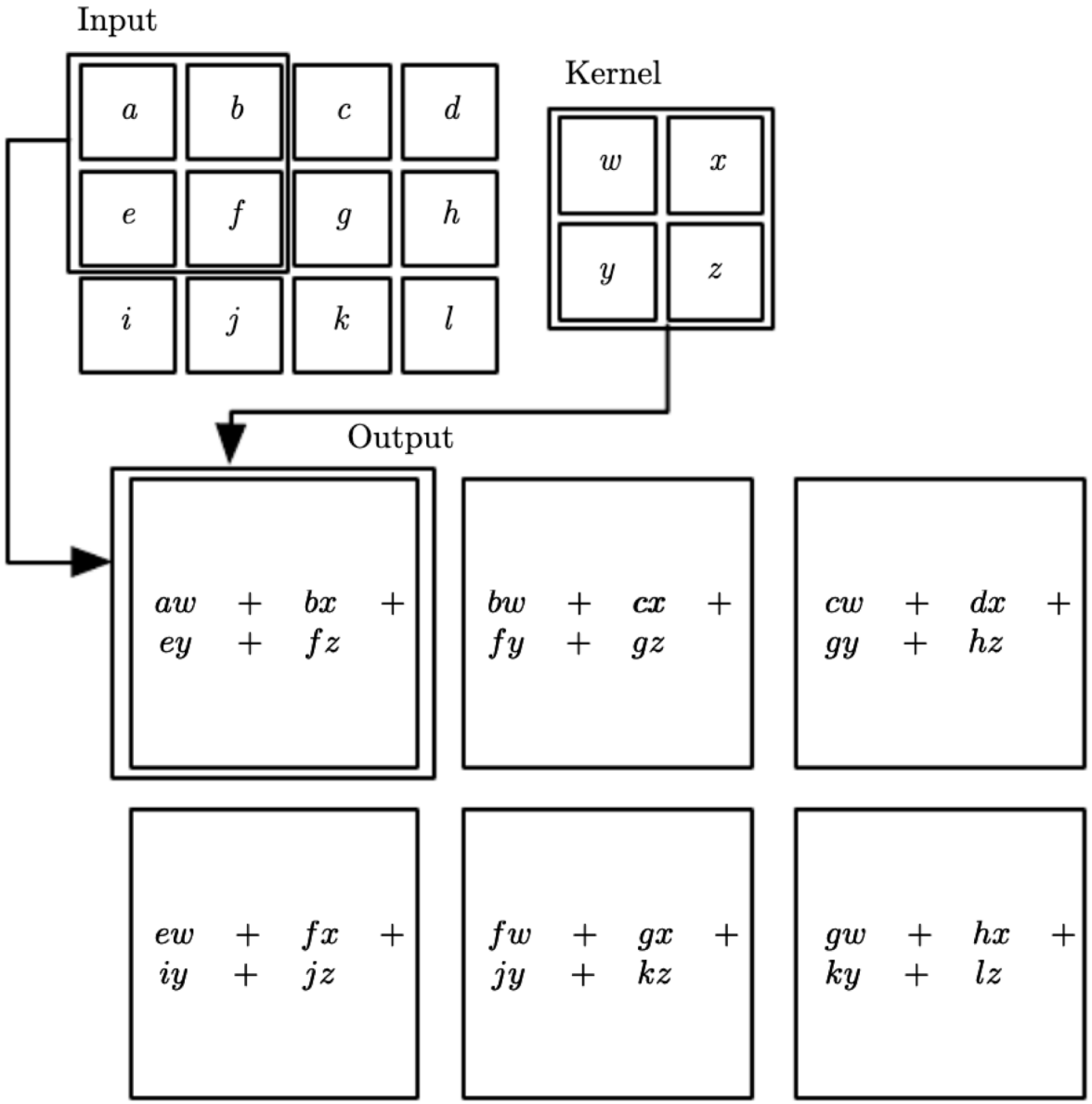
So, a filter applied over an input pattern that is shifted in an image will result in the same convolutional output as the same translational transformation applied after the convolutional—translational equivariance.

https://www.deeplearningbook.org/contents/convnets.html

Convolutional neural networks: 1D convolution

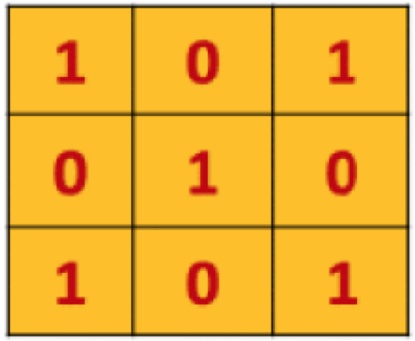


Convolutional neural networks: 2D convolution

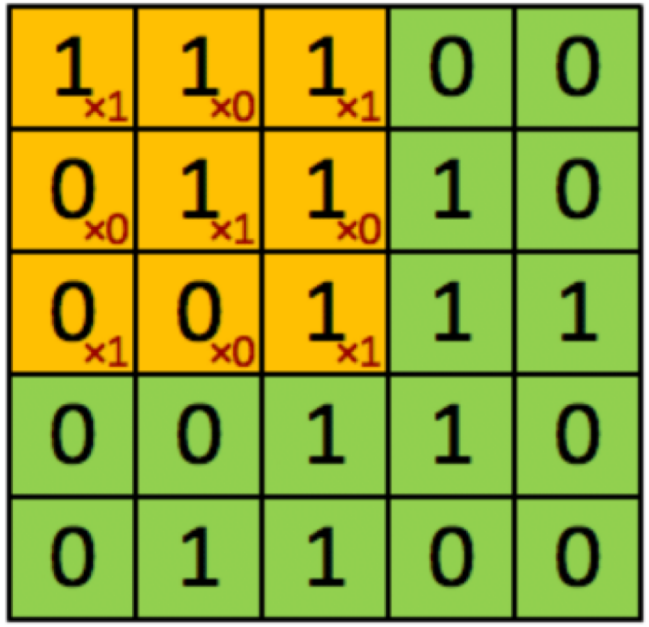


https://www.deeplearningbook.org/contents/convnets.html

Convolutional neural networks: 2D convolution



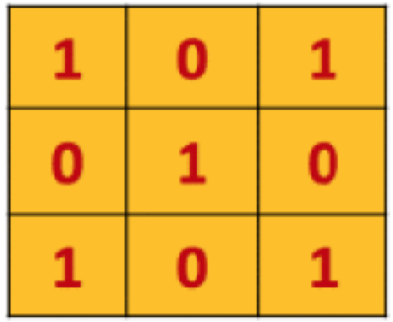
Filter / Kernel



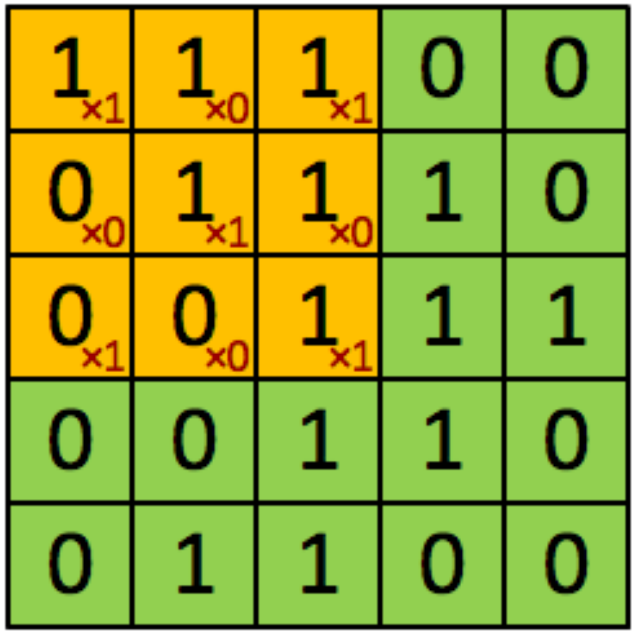
Image

https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

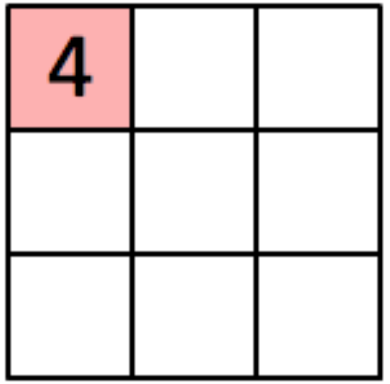
Convolutional neural networks: 2D convolution



Filter / Kernel



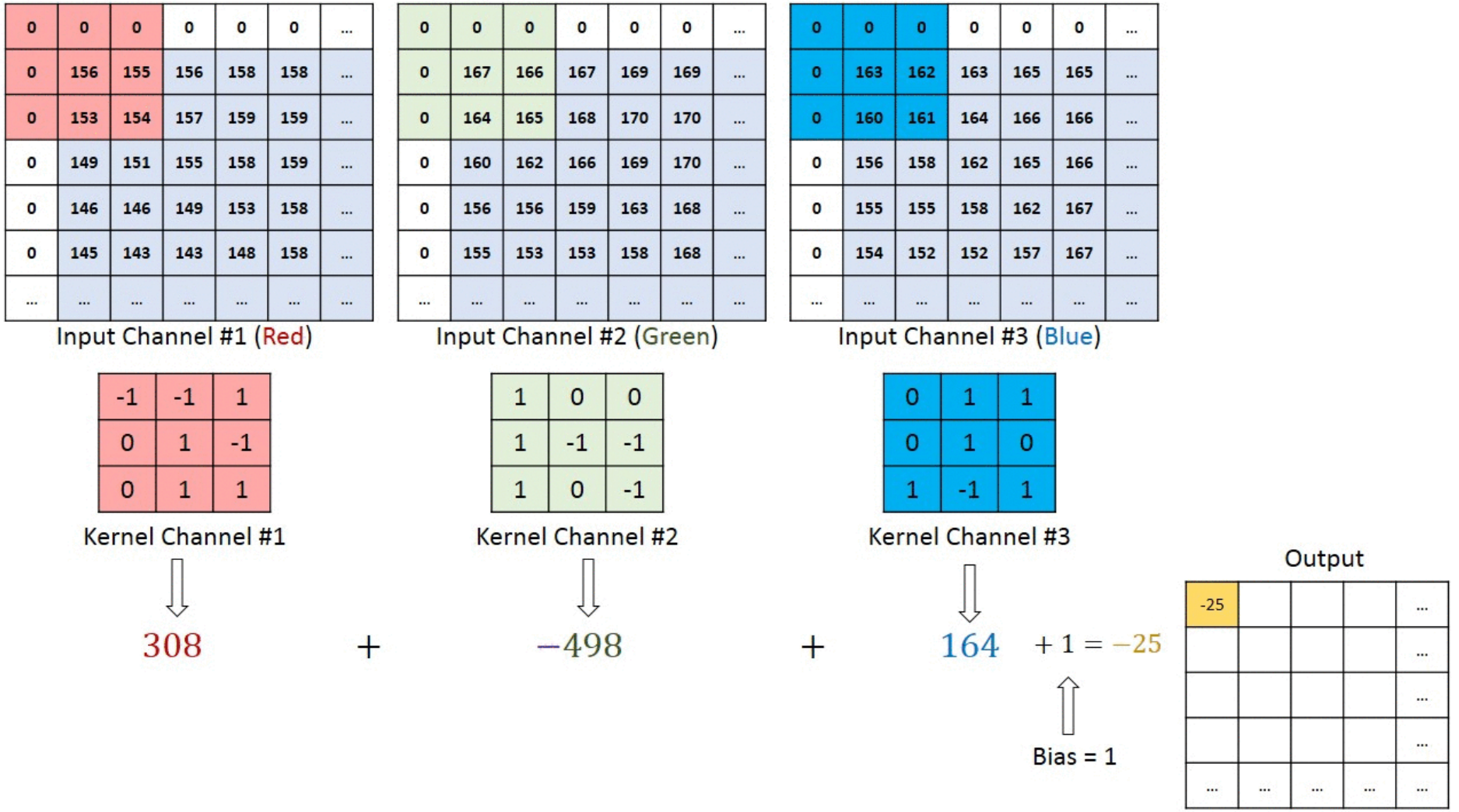
Image



Convolved Feature

https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Convolutional neural networks: multi-channel inputs

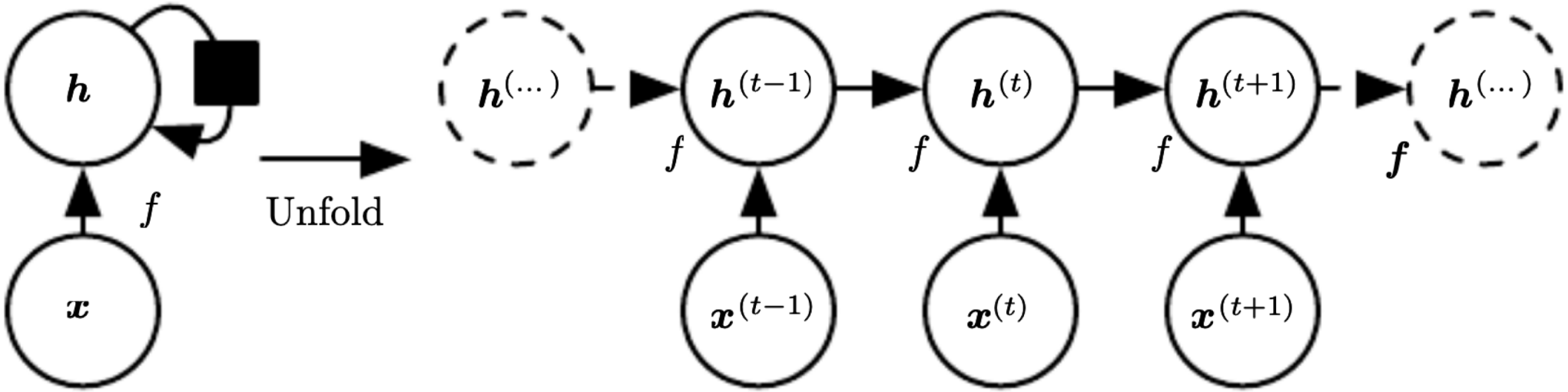


https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Recurrent neural networks: state machine

Recurrence relation w/ external input

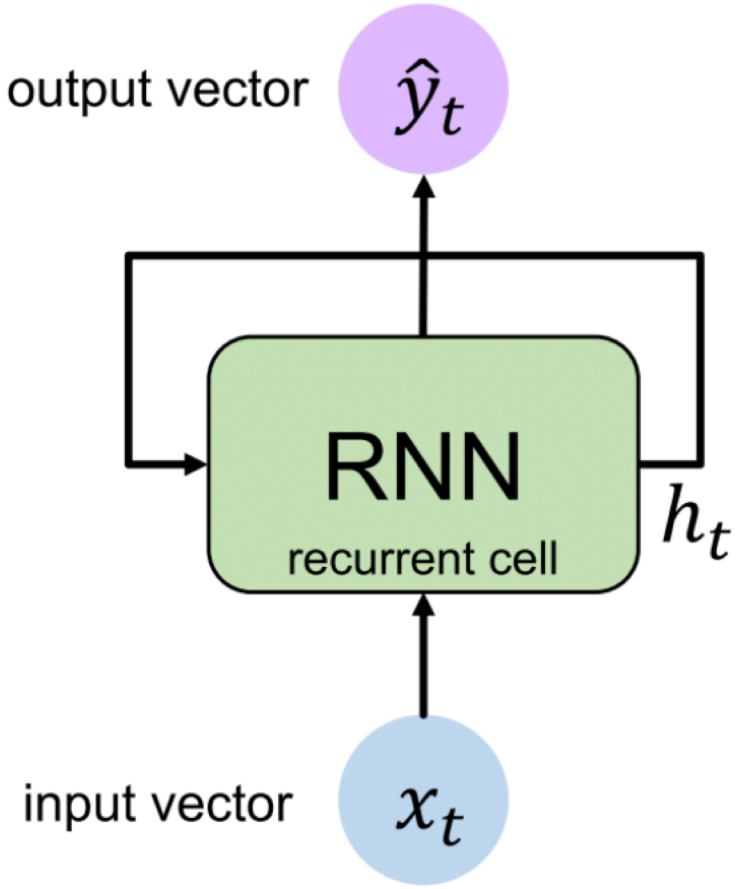
$$\boldsymbol{s}^{(t)} = f\left(\boldsymbol{s}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta}\right)$$



https://www.deeplearningbook.org/contents/rnn.html

Recurrent neural networks: back propagation through time

Idea: Learn sequential / temporal relationships



Apply a **recurrence relation** at every time step to process a sequence:

$$\boxed{h_t} = \boxed{f_W}\left(\boxed{h_{t-1}}, \boxed{x_t}\right)$$

new state — function parameterized by W — old state — input vector at time step *t*

Note: the same function and set of parameters are used at every time step

Model Interpretability

Slide credit to – Dylan Cable + Previous 6.874 Slides

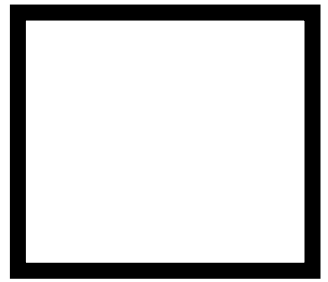
Model interpretability: model interpretation overview

- Adoption of deep learning has led to:
  - Large increase in predictive capabilities
  - Complex and poorly-understood black-box models

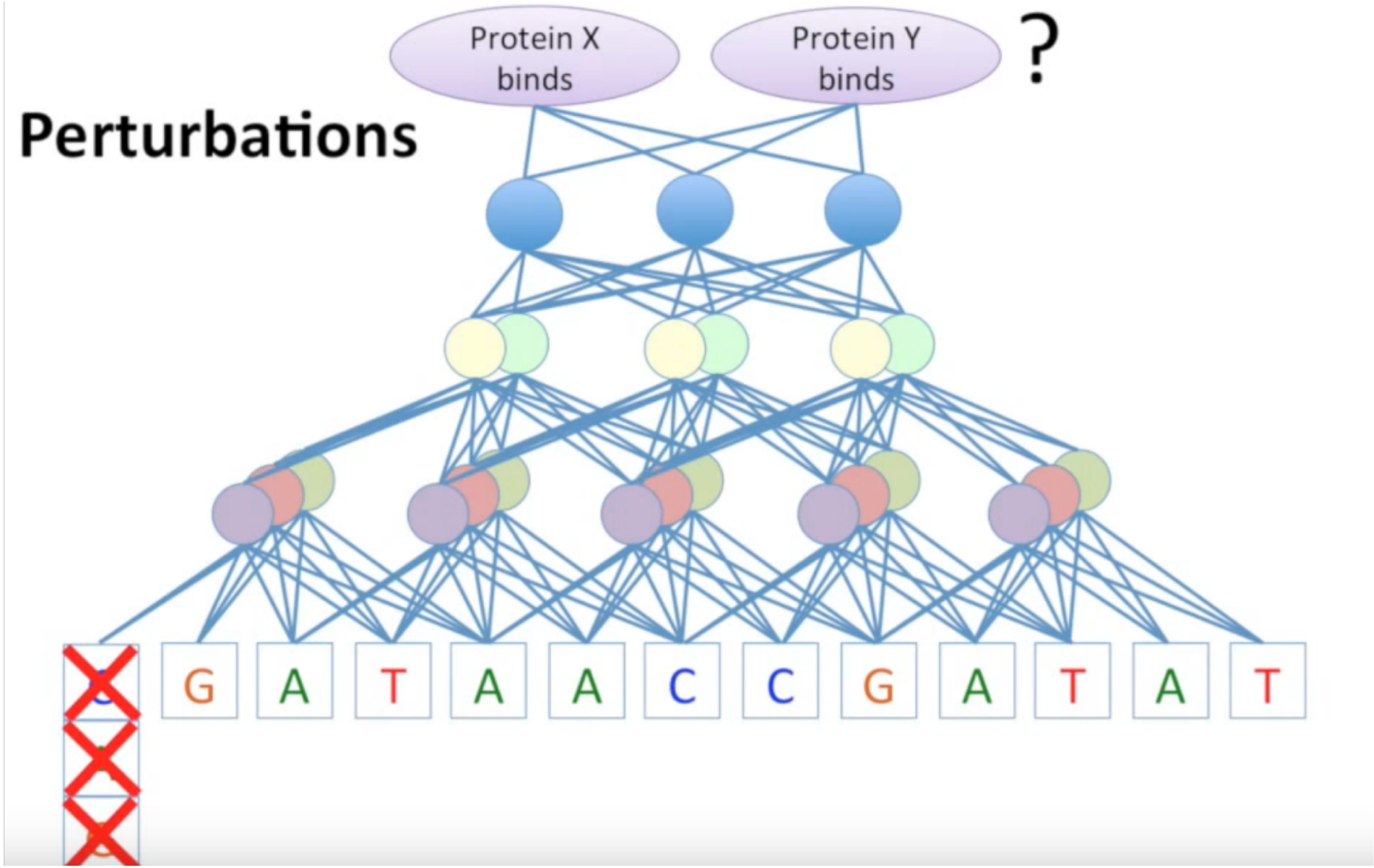
- Imperative that certain model decisions can be interpretably rationalized
  - Ex: loan-application screening, recidivism prediction, medical diagnoses, autonomous vehicles

- Explain model failures and improve architectures

- Interpretability is also crucial in scientific applications, where goal is to identify general underlying principles from accurate predictive models



Model interpretability: black-box models



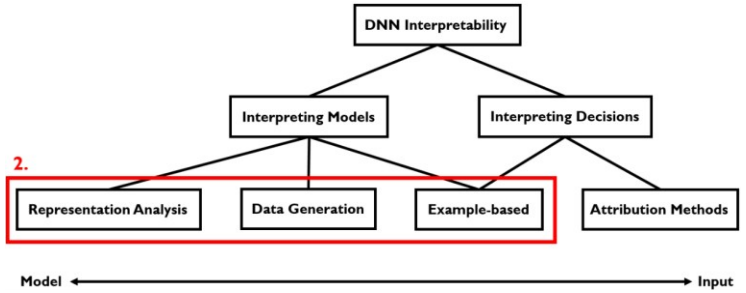
Model Interpretability: Four Basic Strategies

- Visualize the filters
- Measure the Gradient of node with respect to the input (Saliency Map)
- Choose an input that maximizes a node
- Perturb Input and See the effect on output
- Others!

Model Interpretability: Four Basic Strategies

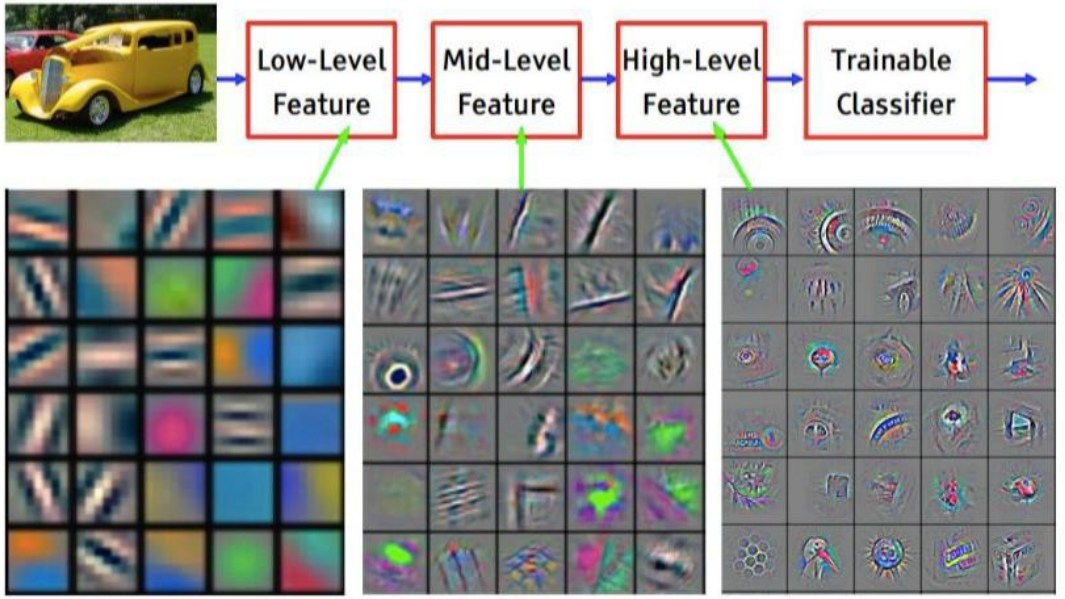
- **Visualize the filters**
- Measure the Gradient of node with respect to the input (Saliency Map)
- Choose an input that maximizes a node
- Perturb Input and See the effect on output
- Others!

Types of DNN Interpretability



WeightVisualization | **Surrogate Model** | **Data Generation** | **Example-based**

- Filter visualization in Convolutional Neural Networks
- Can understand what kind of features CNN has learned
- Still too many filters!



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

"Deep Inside Convolutional Networks: Visualizing Image Classification Models and Saliency Maps", https://arxiv.org/pdf/1312.6034.pdf

Model interpretability: saliency maps

Visualizing saliency

The procedure is related to back-propagation, but in this case the optimization is performed with respect to the input image, while the weights are fixed to those found during the training stage.

More concretely,

$$S_c(I) \approx W^T I + b$$

Where,

$$W = \left.\frac{\partial S_c}{\partial I}\right|_{I_i}$$

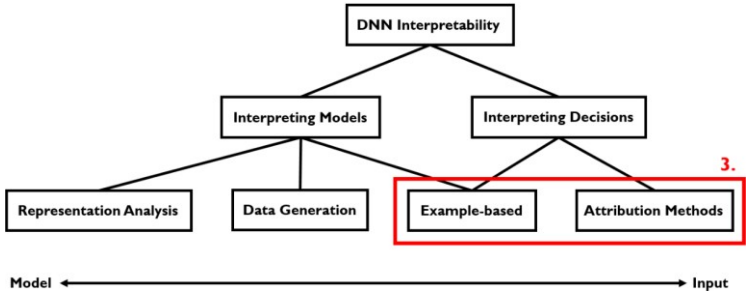
Optimization by gradient ascent

More formally, let Sc(I) be the score of the class c, computed by the classification layer of the ConvNet for an image I. We would like to find (via backprop) an L2-regularised image, such that the score Sc is high,

$$\arg\max_I S_c(I) - \lambda \|I\|_2^2$$

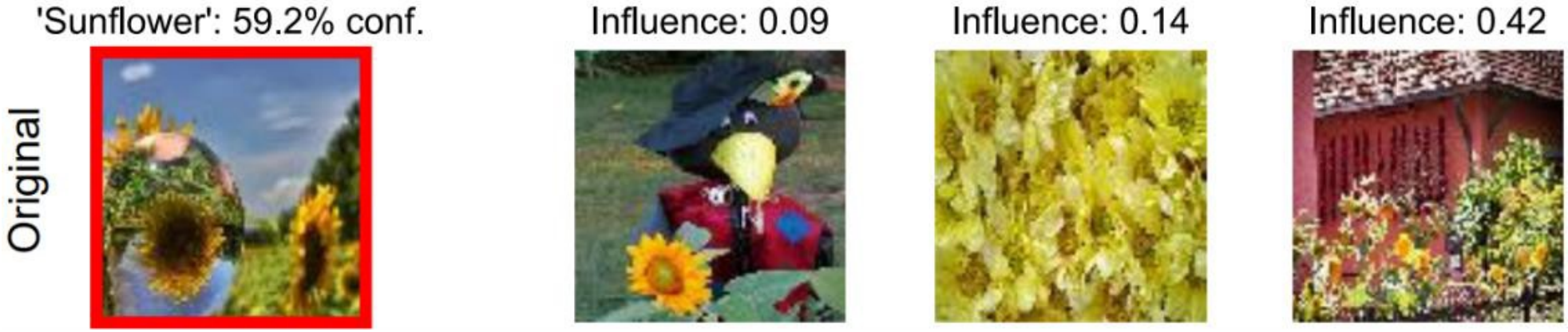
https://arxiv.org/pdf/1312.6034.pdf

Types of DNN Interpretability



Example-based | **Attribution Methods** | **Gradient Based** | **Backprop. Based**

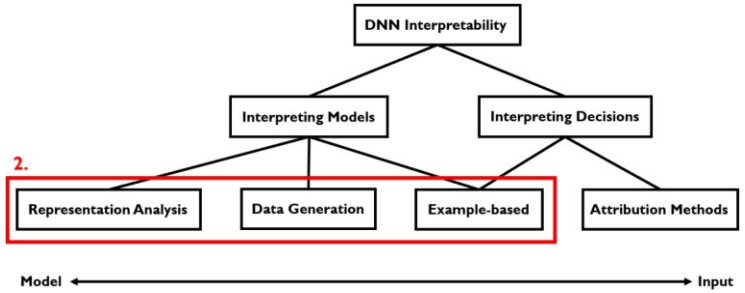
- Which training instance influenced the decision most?
- Still does not specifically highlight which features were important
- Influence functions for interpreting black-box methods. Fragility of NN model interpretation.



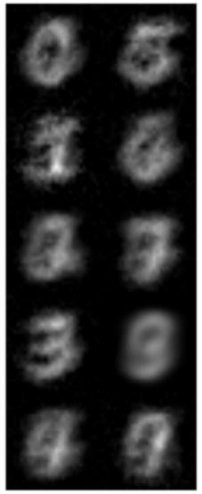
"Understanding Black-box Predictions via Influence Functions", https://arxiv.org/pdf/1703.04730.pdf
"Interpretation of Neural Networks is Fragile", https://arxiv.org/pdf/1710.10547.pdf

Slides by Beomsu Kim, KAIST

Types of DNN Interpretability



WeightVisualization | **Surrogate Model** | **Data Generation** | **Example-based**

	simple AM (initialized to mean)	simple AM (init. to class means)	AM-density (init. to class means)	AM-gen (init. to class means)	
Original Activation Maximization					*Constrained Activation Maximization*

Observation: Connecting to the data leads to sharper visualizations.