

## Lecture 1: Convolutional Neural Networks

Lecturer: Thomas Hofmann

Scribes: Yao Zhang

**Definition 1.1** (Integral operator). A transform  $T$  expressible with the kernel  $H$  and  $t_1, t_2 \in \mathbb{R} \cup \{-\infty, \infty\}$  such that for any function  $f$  (for which  $Tf$  exists)

$$(Tf)(u) = \int_{t_1}^{t_2} H(u, t) f(t) dt \quad (1.1)$$

is called an integral operator.

**Example 1.1** (Fourier transform).

$$(\mathcal{F}f)(u) \triangleq \int_{-\infty}^{\infty} e^{-2\pi i t u} f(t) dt \quad (1.2)$$

**Definition 1.2** (Convolution). Given two functions  $f, h$ , their convolution is defined as

$$(f * h)(u) \triangleq \int_{-\infty}^{\infty} h(u - t) f(t) dt = \int_{-\infty}^{\infty} f(u - t) h(t) dt \quad (1.3)$$

**Remark 1.1.**

1. integral operator with kernel  $H(u, t) = h(u - t)$
2. *shift-invariant* as  $H(u - s, t - s) = h(u - t) = H(u, t) \quad (\forall s)$

Proof. content...

□

3. convolution operator is commutative

Proof. content...

□

4. existence depends on properties of  $f, h$
5. typical use  $f$  = signal,  $h$  = fast *decaying* kernel function

**Definition 1.3** (Linear transform).  $T$  is linear, if for all functions  $f, g$  and the scalars  $\alpha, \beta$ ,

$$T(\alpha f + \beta g) = \alpha Tf + \beta Tg \quad (1.4)$$

**Definition 1.4** (Translation invariant transform).  $T$  is translation (or shift) invariant, if for any  $f$  and scalar  $\tau$ ,

$$f_\tau(t) \triangleq f(t + \tau), \quad (Tf_\tau)(t) \triangleq (Tf)(t + \tau) \quad (1.5)$$

**Remark 1.2.** content...

**Theorem 1.1.** Any linear, translation-invariant transformation  $T$  can be written as *convolution* with a suitable  $h$ .

Proof. content...

□

Signal processing with neural networks:

1. Transforms in deep networks: **linear** + simple non-linearity
2. Many signals (audio, image, etc.) obey **translation invariance**  $\Rightarrow$  **invariant** feature maps: shift in input = shift in feature map

**1** + **2** in above:

1.  $\Rightarrow$  learn convolutions, not (full connectivity) weight matrices
2.  $\Rightarrow$  **convolutional layers** for signal processing

For all practical purposes: signal are sampled, i.e. discrete.

**Definition 1.5** (Discrete convolution (1-D)). For  $f, h : \mathbb{Z} \rightarrow \mathbb{R}$ , we can define the discrete convolution via

$$(f * h)[u] \triangleq \sum_{t=-\infty}^{\infty} f[t] h[u-t] \quad (1.6)$$

**Remark 1.3.**

1. use of rectangular brackets to suggest "arrays"

2. 2D case:

content... (1.7)

3. typical:  $h$  with finite support (window size)

**Example 1.2.** Small Gaussian kernel with support  $[-2 : 2] \subset \mathbb{Z}$

$$h[t] = \frac{1}{16} \begin{cases} 6 & t = 0 \\ 4 & |t| = 1 \\ 1 & |t| = 2 \\ 0 & \text{otherwise} \end{cases} \quad (1.8)$$

Consequence: convolution sum can be truncated:

$$\begin{aligned} (f * h)[u] &= \sum_{t=u-2}^{u+2} f[t] h[u-t] = \sum_{t=-2}^2 h[t] f[u-t] \\ &= \frac{6f[u] + 4f[u-1] + 4f[u+1] + f[u-2] + f[u+2]}{16} \end{aligned} \quad (1.9)$$

**Remark 1.4.** content...

**Definition 1.6** (Discrete cross-correlation). Let  $f, h : \mathbb{Z} \rightarrow \mathbb{R}$ , then

$$(h \otimes f)[u] \triangleq \sum_{t=-\infty}^{\infty} h[t] f[u+t] \quad (1.10)$$

**Remark 1.5.**

1. Def. 1.6 also called a "sliding inner product",  $u + t$  instead of  $u - t$
2. note that cross-correlation and convolution are closely related:

$$\begin{aligned}
 (h \otimes f)[u] &= \sum_{t=-\infty}^{\infty} h[t] f[u+t] \\
 &= \sum_{t=-\infty}^{\infty} h[-t] f[u-t] \\
 &= (\bar{h} * f)[u] \\
 &= (f * \bar{h})[u]
 \end{aligned} \tag{1.11}$$

where  $\bar{h}[t] \triangleq h[-t]$ .

Only difference: kernel flipped over, but not non-commutative.

Convolution via matrices:

1. In practice: signal  $f$  and kernel  $h$  have finite support
2. Without loss of generality (w.l.o.g)  $f[t] = 0$  for  $t \notin [1 : n]$ ,  $h[t] = 0$  for  $t \notin [1 : m]$
3. We can think of  $f$  and  $h$  as vectors and define:

$$\underbrace{\begin{bmatrix} h_1 & 0 & 0 & \cdots & 0 & 0 & \cdots & \cdots & \cdots & \cdots \\ h_2 & h_1 & 0 & \cdots & 0 & 0 & \vdots & \vdots & \vdots & \vdots \\ h_3 & h_2 & h_1 & \cdots & 0 & 0 & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ h_m & h_{m-1} & h_{m-2} & \cdots & h_1 & 0 & \vdots & \vdots & \vdots & \vdots \\ 0 & h_m & h_{m-1} & \cdots & h_2 & h_1 & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & h_m & \cdots & h_3 & h_2 & \cdots & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & h_m & h_{m-1} & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & h_m \end{bmatrix}}_{\triangleq H_n^h \in \mathbb{R}^{(m+n-1) \times n}} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_{m-1} \\ f_m \\ \vdots \\ \vdots \\ f_{n-2} \\ f_{n-1} \\ f_n \end{bmatrix} \tag{1.12}$$

**Remark 1.6.** content...

**Definition 1.7** (Toeplitz matrix). A matrix  $H \in \mathbb{R}^{k \times n}$  is a Toeplitz matrix, if there exists  $n+k-1$  numbers  $c_l$  ( $l \in [-(n-1) : (k-1)] \subset \mathbb{Z}$ ) such that

$$H_{ij} = c_{i-j} \tag{1.13}$$

**Remark 1.7.**

1. in plain English, all **NW-SE** diagonals are constant
2. if  $m \ll n$ : additional sparseness (band matrix of width  $m$ )

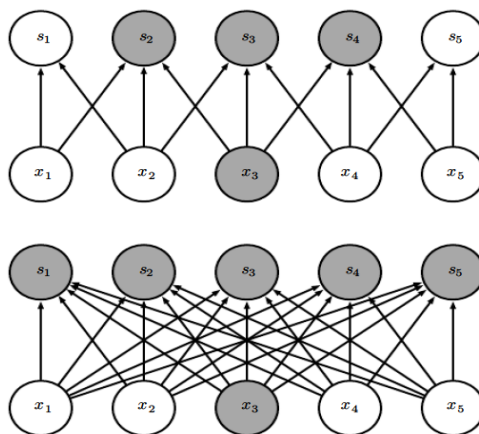


Figure 1.1: Sparse vs dense connectivity

3.  $H_n^h$  has only  $m$  degrees of freedom
4. locality (sparseness  $m \ll n$ ) and weight sharing (kernel)

Convolutions in higher dimensions: generalize concept of convolution to:

1. 2D: e.g. images, spectrograms
2. 3D: e.g. color or multi-spectral images, voxel images, video
3. or even higher dimensions

Replace vector by:

1. matrices or fields (e.g. in discrete case)

$$(F * G)[i, j] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} F[i - k, j - l] \cdot G[k, l] \quad (1.14)$$

2. tensors: for 3D and higher

Different options for border handling:

1. our definition: padding with zeros = **same padding**
2. only retain values from windows fully contained in support of signal  $f$  = valid padding

layout:

1. Convolved signal **inherits** topology of original signal
2. Hence: units in a convolutional layer are typically arranged on the same grid (1D, 2D, 3D,...)

Exploit **structural sparseness** in computing  $\frac{\partial x_i^l}{\partial x_j^{l-1}}$ :

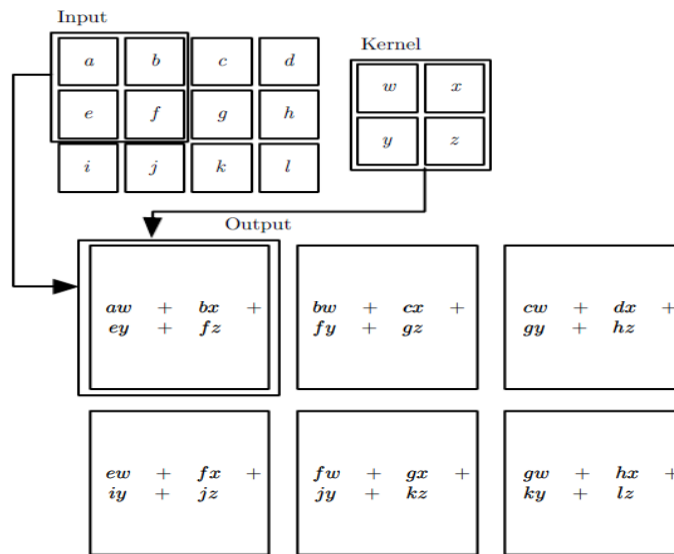


Figure 1.2

1. receptive field of  $x_i^l : \mathcal{I}_i^l \triangleq \{j : W_{ij}^l \neq 0\}$ , where  $W^l$  is the Toeplitz matrix of the convolution
2. obviously  $\frac{\partial x_i^l}{\partial x_j^{l-1}} = 0$  for  $j \notin \mathcal{I}_i^l$

**Weight sharing** in computing  $\frac{\partial \mathcal{R}}{\partial h_j^l}$ , where  $h_j^l$  is a kernel weight

$$\frac{\partial \mathcal{R}}{\partial h_j^l} = \sum_i \frac{\partial \mathcal{R}}{\partial x_i^l} \frac{\partial x_i^l}{\partial h_j^l} \quad (1.15)$$

Weight is re-used for every unit within target layer  $\Rightarrow$  additive combination of derivatives in chain rule.

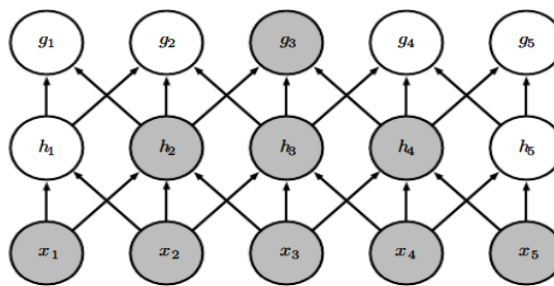


Figure 1.3

FFT (**F**ast **F**ourier **T**ransform): compute convolutions fast(er).

1. Fourier transform of signal  $f \rightarrow (\mathcal{F}f)$  and kernel  $h \rightarrow (\mathcal{F}h)$
2. pointwise multiplication and inverse Fourier transform:

$$(f * h) = \mathcal{F}^{-1}((\mathcal{F}f) \cdot (\mathcal{F}h)) \quad (1.16)$$

3. FFT: signal of length  $n$ , can be done in  $O(n \log n)$
4. pays off, if many channels (amortizes computation of  $\mathcal{F}f$ )
5. small kernels ( $m < \log n$ ): favor time / space domain

**Remark 1.8.** *content...*

Stages:

1. **Non-linearities**: detector stage. As always: scalar non-linearities (activation function)
2. **Pooling** stage: locally combine activities

Most frequently used pooling function: **max pooling**.

**Definition 1.8** (Max Pooling). *Define window size  $r$  (e.g. 3 or  $3 \times 3$ ), then*

$$\begin{aligned} 1D : \quad x_i^{\max} &= \max \{x_{i+k} : 0 \leq k < r\}, \\ 2D : \quad x_{ij}^{\max} &= \max \{x_{i+k, j+l} : 0 \leq k, l < r\} \end{aligned} \tag{1.17}$$

**Remark 1.9.**

1. *maximum over a small patch of units*
2. *other functions are possible: average, soft-maximization*

Max-pooling: invariance

1. set of invertible transformations  $\mathcal{T}$  : group w.r.t composition
2.  $\mathcal{T}$ -invariance through maximization  $f_{\mathcal{T}}(x) \triangleq \max_{\tau \in \mathcal{T}} f(\tau x)$

**Proposition 1.1.**  $f_{\mathcal{T}}$  is invariant under  $\tau \in \mathcal{T}$ .

*Proof.*

$$f_{\mathcal{T}}(\tau x) = \max_{\rho \in \mathcal{T}} f(\rho(\tau x)) = \max_{\rho \in \mathcal{T}} (f(\rho \circ \tau)x) = \max_{\sigma \in \mathcal{T}} f(\sigma x) \tag{1.18}$$

as  $\forall \sigma, \sigma = \rho \circ \tau$  with  $\rho = \sigma \circ \tau^{-1}$ . □

sub-sampling(also known as (aka) strides):

1. often, it is desirable to reduce the size of feature maps
2. **sub-sampling**: reduce temporal/spatial resolution. Often: combined with (max-)pooling (aka. stride)
3. example: max-pool, filter  $2 \times 2$ , stride  $2 \times 2$
4. disadvantage: loss of information

Learn multiple convolution kernel (or filters) = multiple **channels**:

1. typically: all channels use same window size

2. channels form additional dimension for next layer (e.g. 2D signal  $\times$  channels = 3D tensor)
3. number of channels: design parameter

<http://cs231n.github.io/assets/conv-demo/index.html>

Note that kernels (across channels) form a linear map:

$$h : \mathbb{R}^{r^2 \times d} \rightarrow \mathbb{R}^k \quad (1.19)$$

where  $r \times r$  is the window size and  $d$  is the depth.

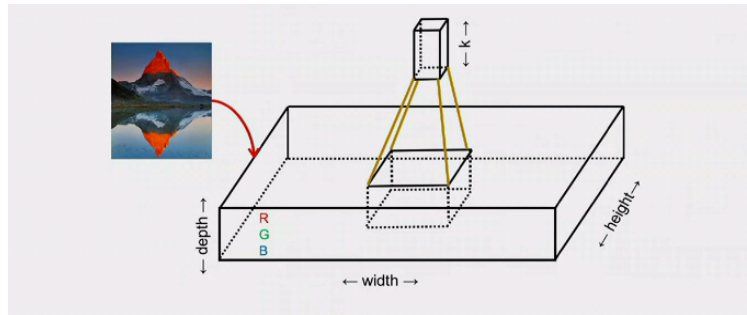


Figure 1.4: convolutional layers for vision

Convolutional networks: **multiple, stacked feature maps**

$$\underbrace{y[r]}_{r\text{-th channel}}[s, t] = \sum_u \sum_{\Delta s, \Delta t} \underbrace{w[r, u][\Delta s, \Delta t]}_{\text{parameters}} \underbrace{x[u]}_{u\text{-th channel}}[s + \Delta s, t + \Delta t] \quad (1.20)$$

1.  $x, y$  tensor, 3-rd order
2. number of parameters:

$$\underbrace{\#r \cdot \#u}_{\text{fully connected}} \cdot \underbrace{\#\Delta s \cdot \#\Delta t}_{\text{window size}} \quad (1.21)$$

3. pointwise non-linearities (e.g. ReLU)
4. interleaved with: pooling (e.g. max, average)
5. optionally: downsampling (use of strides)

Convolutional pyramid:

Typical use of convolution in vision: sequence of convolutions that

1. **reduce** spatial dimensions (sub-sampling)
2. **increase** number of channels

$\Rightarrow$  smaller, but more feature maps.

LeNet5 [1,2]

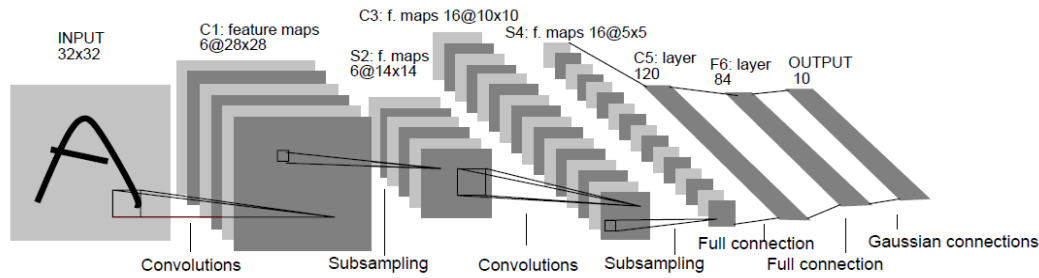


Figure 1.5: Architecture of LeNet-5, a convolutional neural network, here for digits recognition. Each plan is a feature map, i.e. a set of units whose weights are constrained to be identical. [2]

1. C1/S2: 6 channels,  $5 \times 5$  kernels,  $2 \times 2$  sub (4704 units)
2. C3/S4: 16 channels,  $6 \times 6$  kernels,  $2 \times 2$  sub (1600 units)
3. C5: 120 channels, F6: fully-connected
4. output: Gaussian noise model (squared loss)

#### AlexNet[3]

1. **Pyramidal architecture**: reduce spatial resolution, increase channels with depth
2. Challenge: many channels (width) + large windows + depth
3. Number of parameters
  - (a) 384 to 384 channels with  $3 \times 3$  windows:  $> 1.3$  M
  - (b)  $13 \times 13 \times 384$  tensor to 4096, fully connected:  $> 265$  M

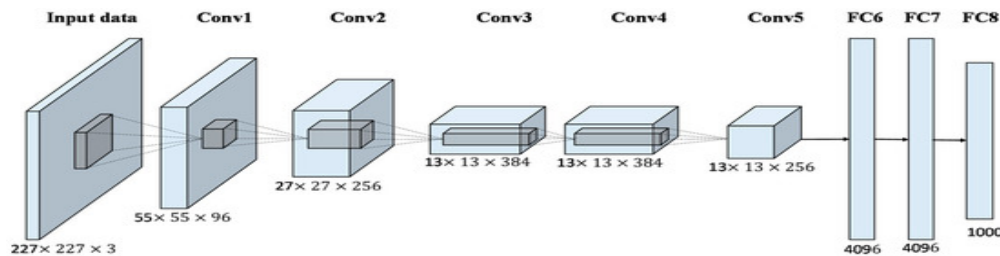


Figure 1.6: AlexNet architecture

#### Deep ConvNets: key challenges

1. avoid blow-up of model size (e.g. # parameters)
2. preserve computational efficiency of learning (e.g. gradients)
3. allow for large depth (as it is known to be a plus)
4. allow for sufficient width (as it is known to be a plus, too)



## Very deep convolutional networks: VGG [4]

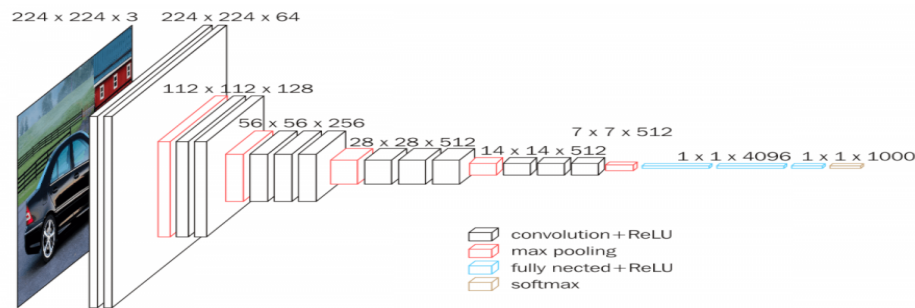


Figure 1.7: VGG 16

1. use very small receptive fields (maximally  $3 \times 3$ )
2. avoid downsampling/pooling
3. stacking small receptive fields: more depth, fewer parameters
4. example:  $3 \cdot (3 \times 3) = 27 < 49(7 \times 7)$

**Many channels** needed for high accuracy, typically  $k \sim 200 - 1000$  (e.g. AlexNet:  $2 \times 192$ ).

Observation (motivated by Arora et al, 2013 [5]): when convolving, dimension reduction across channels may be acceptable.

**Dimension reduction:**  $m$  channels of a  $1 \times 1 \times k$  convolution  $m \leq k$ :

$$x_{ij}^+ = \sigma(Wx_{ij}), \quad W \in \mathbb{R}^{m \times k} \quad (1.22)$$

1.  $1 \times 1$  convolution = no convolution
2. inception module (Szegedy et al. [6])
3. network within a network (Lin et al, [7])
4. i.e.  $W$  is shared for all  $(i, j)$  (translation invariance)

Inception module: mixing

Instead of fixed window size convolution: **mix**  $1 \times 1$  with  $3 \times 3$  and  $5 \times 5$ , max-polling. Use  $1 \times 1$  convolutions for dimension reduction before convolving with large kernels.

Google inception networks [6]

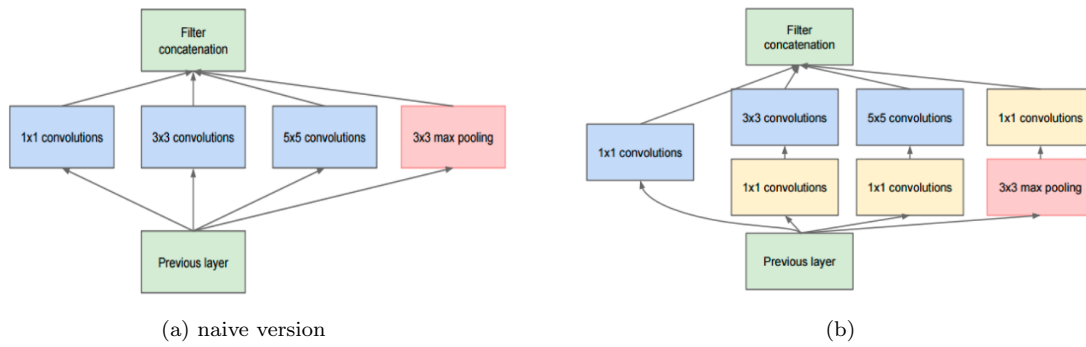


Figure 1.8: Inception module [8]

Very deep network: many inception modules (green boxes: concatenation points). Additional trick: connect softmax layer (and loss) at intermediate stages (yellow boxes)  $\Rightarrow$  gradient shortcuts.



Figure 1.9: Google inception networks

Residual networks: ResNets [9]

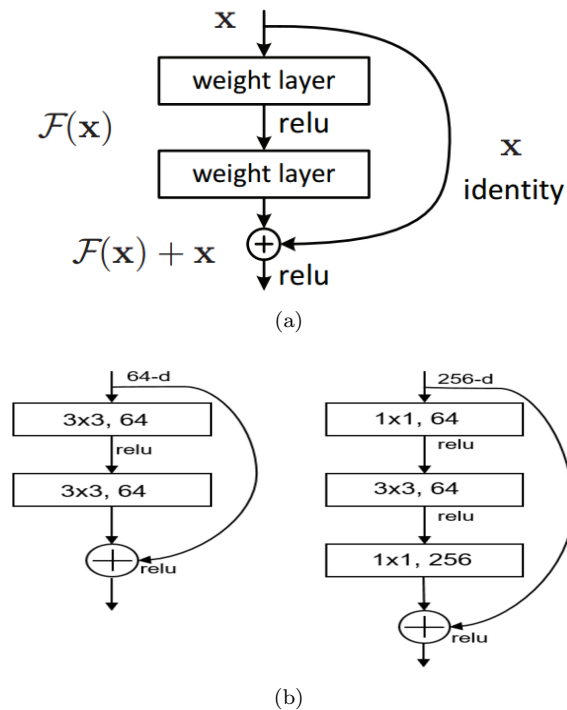


Figure 1.10: Residual Networks module [9]

1. learn changes to the **identity map** (aka. shortcut connections)
2. use small filters (VGG), use dimension reduction (inception)
3. reach depth of 100 + layers (+ increase accuracy + trainable)

Next topic is convolutional sequence models and recurrent networks

## Reading List

- [1] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard and L. Jackel, "Back-propagation applied to handwritten zip code recognition," *Neural Computation*, 1989, Vol. 1(4), pp. 541–551.
- [2] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, 1998, Vol. 86(11), pp. 2278–2324.
- [3] A. Krizhevsky, I. Sutskever and G. Hinton, "ImageNet classification with deep convolutional neural networks," *NIPS 2012, Neural Information Processing Systems*, 2012, Vol. 60, pp. 84–90.
- [4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *International Conference on Learning Representations*, 2015.

- [5] M. Arora and H. Kaur, “Performance analysis of communication system with convolutional coding over fading channel,” *International Journal of Scientific & Engineering Research* , 2013, Vol. 4(5), pp. 1116–1120.
- [6] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhanand, V. Vanhoucke and A. Rabinovich, “Going deeper with convolutions,” *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* , 2015, pp. 1–9.
- [7] M. Lin, Q. Chen, S. Yan, “Network in network,” *International Conference on Learning Representations (ICLR)*, 2013.
- [8] C. Vasconcelos, B. Vasconcelos, “Network in convolutional neural network committees for melanoma classification with classical and expert knowledge based image transforms data augmentation,” *arXiv:1702.07025*, 2017.
- [9] K. He, X. Zhang, S. Ren and J. Sun, “Deep Residual Learning for Image Recognition,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* , 2016, pp. 770–778.