The department of Mechanical Engineering
National Taiwan University

Week1 Report

B12502066 許慈方

2025 summer seminar - IEV&M Lab

Table of Contents

Chapter1 Convolution Neural Networks

1.1 Convolution layer (core building block of CNN)

1.1.1 Input data: color image made up of a matrix of pixels in 3D and also RGB

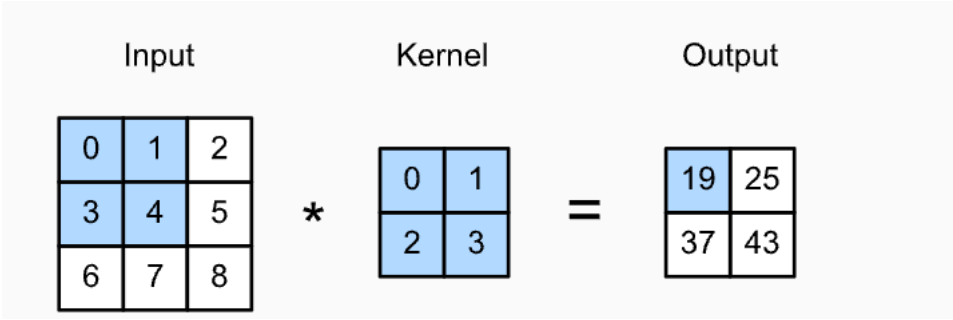
1.1.2 Filter or Kernel (feature detector): A small matrix of weights that detects specific features.

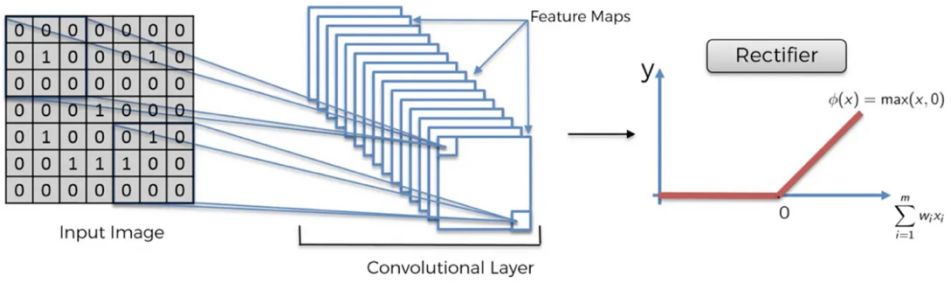
1.1.1.1 Number of filters: affects the depth of the filter

1.1.1.2 Stride: the distance or number of pixels that the kernel moves over the matrix. A larger stride yields a smaller output.

1.1.3 Feature map: output array

Working principle: When the filter applies to the image, a dot product is calculated between the input pixels and the filter. The dot product then fed into an output array. After the filter shifts by a stride, repeating the process until the kernel has swept across the entire image. The final output is known as feature map.





1.2 Pooling layer

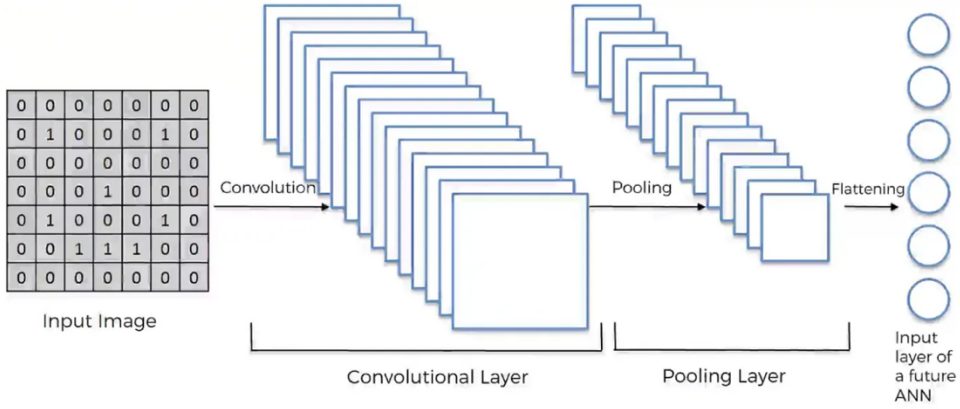
Conducts dimensionality reduction, reducing the number of parameters in the input. The filter of pooling layer does not have trainable weights, but it still sweeps across the input.

1.2.1 Max pooling: selects the pixel with maximum filter value to send to the output array.

1.2.2 Average pooling: calculates the average value within the receptive field to. send to the output array

1.2 Fully Connected Layer

Each node in the output layer connects directly to a node in a previous layer. It performs classification based on features extracted through layers and different filters.



Chapter 2 Multi-layer perceptron

2.1 Multi-layer perceptron

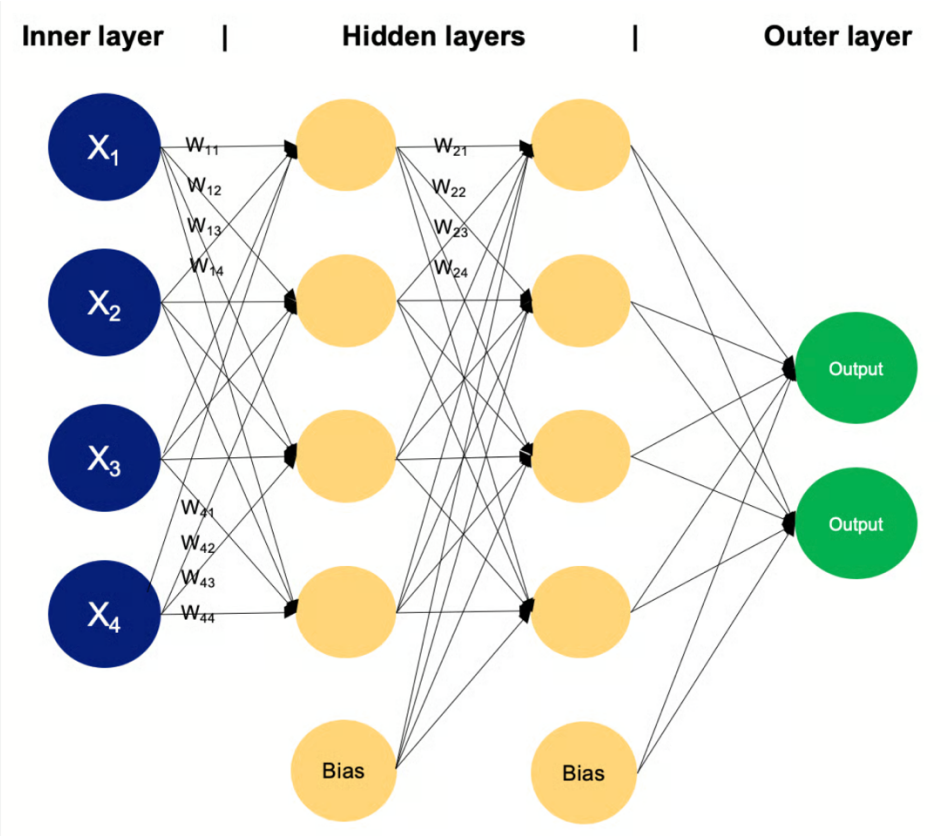
Multi-layer Perceptron contains an input layer, one or more hidden layers and an output layer. Using fully connected dense layers to transform input data from one dimension to another, in order to model complex relationships between input and output.

Fully connected nature: Every node in one layer connects to every node in the next layer.

2.1.1 Input layer: Each neuron or node corresponds to an input feature.

2.1.2 Hidden layers: These layers process the information received from the input layer.

2.1.3 Output layer: Generates the final result, it has corresponding number of neurons.



2.2 Key mechanisms

2.2.1 Forward Propagation: data flows from the input layer to the output layer

2.2.1.1 Weighted Sum: $\sum_i w_i x_i + b$ $(x_i: input\ feature\ w_i: weight\ b: bias)$

2.2.1.2 Activation Function: The weighted sum z is passed through an activate function to introduce non-linearity.

1. Sigmoid: $\sigma(z) = \frac{1}{1+e^{-z}}$

2. ReLU (Rectified Linear Unit): $f(z) = \max(0, z)$

3. Tanh: $\tanh(z) = \frac{2}{1+e^{-2z}} + 1$

2.2.2 Loss function

A function that measures the difference between predicted output and actual target. Common loss functions include Mean Squared Error (MSE) and Cross Entropy.

$$L = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

Where:

- y_i is the actual label.
- $\hat{y}_i$ is the predicted label.
- N is the number of samples.

2.2.3 Backpropagation

2.2.3.1 Gradient calculation: The gradients of the loss function respect to each weight and bias are calculated using the chain rule.

2.2.3.2 Error propagation: The error propagated back through the network, layer by layer.

2.2.3.3 Gradient descent: The network updates the weights and biases by moving in the opposite direction of the gradient to reduce the loss:

$$\omega = \omega - \eta \cdot \frac{\partial L}{\partial \omega}$$

(η : learning rate, $\frac{\partial L}{\partial \omega}$: gradient of loss function with respect to the weight)

2.2.4 Optimization

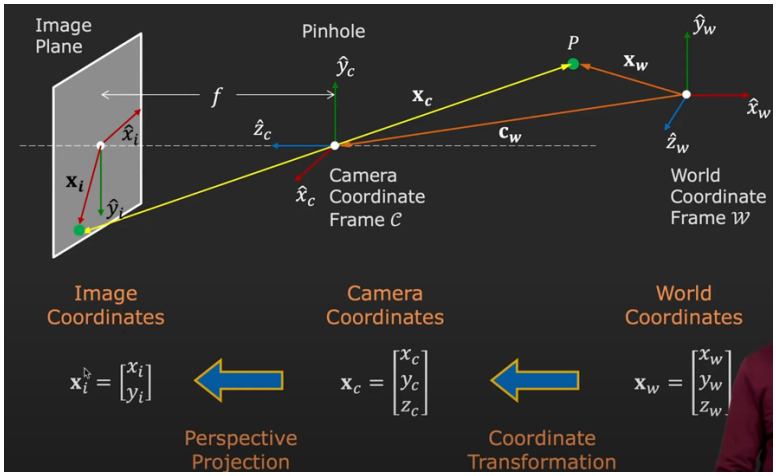
2.2.4.1 Stochastic Gradient Descent (SGD): Updates the weights based on a single sample or a small batch of data.

2.2.4.2 Adam Optimizer: An extension of SGD that incorporates momentum and adaptive learning rates for more efficient training.

Chapter3 Camera model

Camera model mathematically describes how 3D points in the real world are projected onto a 2D image captured by a camera.

$$m = K[R|T]W \quad \rightarrow \quad \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$



3.1 Extrinsic parameters (world coordinates to camera coordinates)

Extrinsic parameters: Position C_w, rotation matrix (R), translation vector (t) in the world coordinate frame. Rotation matrix R is orthonormal.

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = R \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

3.2 Perspective Projection

f_x and f_y are focal lengths in pixels in x and y directions, then the pixel coordinates are:

$$\begin{cases} u = f_x \dfrac{x_c}{z_c} + o_x \\ v = f_y \dfrac{y_c}{z_c} + o_y \end{cases}$$

3.2.1 Homogenous coordinates of (u, v)

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x x_c + z_c o_x \\ f_y y_c + z_c o_y \\ z_c \end{bmatrix} = \begin{bmatrix} f_x & 0 & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix}$$

3.3 Intrinsic Matrix (camera coordinates to image coordinates)

Intrinsic parameters: f_x, f_y(focal lengths in pixels), o_x, o_y (principle points) that represent the camera's internal geometry. The intrinsic matrixM_{int} is the calibration matrix k expressed in homogeneous coordinates.

Calibration matrix: $k = \begin{bmatrix} f_x & 0 & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix}$ (upper right triangle matrix)

Intrinsic matrix: $M_{int} = [k|0]\begin{bmatrix} f_x & 0 & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$

3.4 Distortion coefficients (Real lenses)

This model is primarily used to characterize pixel distortion at the periphery of an image induced by curved or wide-angle lenses, mainly comprising radial distortion and tangential distortion.

Chapter 4 Reponses to questions on Resnet Paper

4.1 What are the challenges this paper tries to solve for?

Degradation problem: With the network depth increasing, accuracy gets statured and then degrades rapidly. Adding more layers to deep model leads to higher training error.

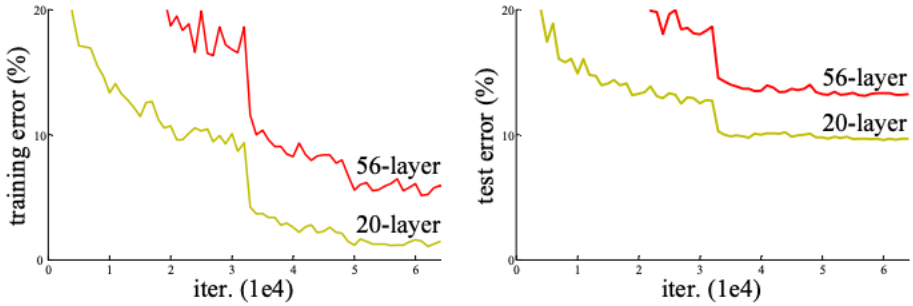


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

4.2 What is the main contribution of this paper? (152-layer residual net)

By designing a deep residual learning architecture, the network effectively mitigates the problems of vanishing and exploding gradients as depth increases. As a result, ResNet not only enhances model accuracy and stability but also enables the training of much deeper neural networks.

4.3 Please understand the details of the network structures.

4.3.1 Deep residual learning framework: Instead of directly learning the desired mapping $H(x)$, the residual learning framework lets the network learn the residual function $F(x) = H(x) - x$. The original mapping becomes $H(x) = F(x) + x$. This design makes it easier to learn, especially when the optimal mapping is close to an identity function, in which case the residual function $F(x)$ approaches zero.

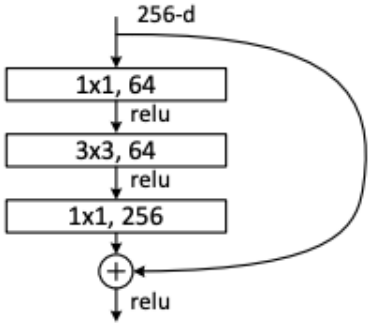
4.3.2 Short connections: When the input x and the output of the residual function $F(x)$ have the same dimensions, the shortcut simply performs an identity mapping by directly adding the input to the output. These connections add neither additional parameters nor computational

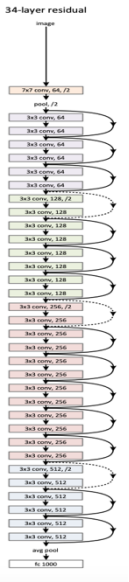
complexity, allowing the network to be trained end to end using SGD with backpropagation.

4.3.3 Projection shortcuts: When input and output dimensions differ, projection shortcuts are used to match dimensions. This is typically done with a 1×1 convolutional layer, denoted as W_s:

$$y = \mathcal{F}(x, \{W_i\}) + W_s x$$

4.3.4 Bottleneck design: The 1 x 1 convolutional layers are utilized to reduce and. subsequently restoring the feature dimensions. It effectively transforms the intermediate 3 x 3 layer into a bottleneck, allowing the module to operate with considerably smaller input and output dimensions.





4.4 Try to find out the current attitude to this paper contribution.

After reviewing several recent papers that utilize ResNet in various deep learning applications, it is evident that the research community still acknowledge the importance of ResNet and continue to improve and apply it in various deep learning tasks. Many studies seek to improve ResNet's performance by integrating additional modules, such as attention mechanisms, and apply the modified architectures across diverse domains including UAV-based object detection, medical imaging, and land use and land cover (LULC) classification. In these applications, attention-enhanced ResNet variants are often employed to improve the focus on target regions and boost classification accuracy.

ResNet's core innovation: residual connections solve the vanishing gradient problem, enhance feature propagation, and support the training of significantly deeper networks

with fewer parameters. The overall attitude in the literature toward ResNet is mostly **positive**, considering it a cornerstone in the development of deep neural networks.

From my perspective, I believe that while ResNet has enabled major advances in deep learning, its potential has yet to be completely realized. Issues such as overfitting in certain applications remain a challenge. Many current approaches rely on modifying and extending ResNet with additional modules rather than using it in its original form. Whether in image processing, visual tracking, or semantic segmentation, ResNet continues to serve as a reliable backbone.

4.5 Notes from papers that mentioned ResNet

1. ConvNet: Transformers lack inductive biases such as locality and translation equivariance that ConvNets naturally provide. Due to their global attention mechanism, transformers suffer from higher computational complexity and greater resource demands, especially with high-resolution inputs. Therefore, by modernizing and enhancing the original ResNet architecture with contemporary design elements, it is possible to achieve improved performance while retaining the efficiency and simplicity of convolutional models.

2. NVIDIA StyleGAN2: The discriminator in StyleGAN2 adopts a residual architecture which leads to consistently improvements in FID and PPL scores. “Skip connections in the generator drastically improve PPL in all configurations, and a residual discriminator network is clearly beneficial for FID. The latter is perhaps not surprising since the structure of discriminator resembles classifiers where residual architectures are known to be helpful.”

3. COVID-Net: COVID-Net leveraged ResNet's residual design principles to build deeper and more reliable initial models. Though the final architecture evolved through human-machine design with unique features like PEPX patterns and long-range connectivity**,** ResNet-50 remained a key benchmark to showcase COVID-Net’s improved efficiency and detection accuracy.

4. Vision Transformer (VIT): ResNet served as a strong convolutional baseline for evaluating the performance of Vision Transformers (ViT). While ViT models slightly underperformed compared to ResNets of similar size when trained on medium-scale datasets like ImageNet. However, when trained on much larger datasets such as JFT-300M, ViT models surpassed ResNet and other CNN-based methods, demonstrating that ViT can achieve better performance under sufficient data and computation.

References

[1] https://www.ibm.com/think/topics/convolutional-neural-networks

[2]https://medium.com/jameslearningnote/%E8%B3%87%E6%96%99%E5%88%86%E6%9E%90-%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92-%E7%AC%AC5-1%E8%AC%9B-%E5%8D%B7%E7%A9%8D%E7%A5%9E%E7%B6%93%E7%B6%B2%E7%B5%A1%E4%BB%8B%E7%B4%B9-convolutional-neural-network-4f8249d65d4f

[3] https://d2l.ai/chapter_convolutional-neural-networks/why-conv.html

[4]https://www.geeksforgeeks.org/deep-learning/multi-layer-perceptron-learning-in-tensorflow/

[5] https://www.datacamp.com/tutorial/multilayer-perceptrons-in-machine-learning

[6]https://www.youtube.com/watch?v=qByYk6JggQU&list=PL2zRqk16wsdoCCLpou-dGo7QQNks1Ppzo&index=2

[7] https://www.sciencedirect.com/topics/engineering/pinhole-camera-model

[8]https://allen108108.github.io/blog/2020/02/06/%E9%87%9D%E5%AD%94%E7%9B%B8%E6%A9%9F%E6%A8%A1%E5%9E%8B%20%20Pinhole%20Camera%20Model/

[9]https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf

[10]https://www.sciencedirect.com/science/article/pii/S2666827021000670(deep learning in computer vision)

[11]https://allen108108.github.io/blog/2019/10/29/[%E8%AB%96%E6%96%87]%20Deep%20Residual%20Learning%20for%20Image%20Recognition/

[12]https://openreview.net/pdf?id=YicbFdNTTy (VIT)

[13]https://www.nature.com/articles/s41598-020-76550-z （Covid-Net）

[14]https://openaccess.thecvf.com/content_CVPR_2020/papers/Karras_Analyzing_and_Improving_the_Image_Quality_of_StyleGAN_CVPR_2020_paper.pdf (StyleGan)

[15]https://openaccess.thecvf.com/content/CVPR2022/papers/Liu_A_ConvNet_for_the_2020s_CVPR_2022_paper.pdf (ConvNet)