SRA
(Live CCTV)

Camlin Page
Date / /

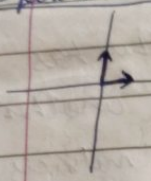
→ CS student: Vectors ⟺ lists of numbers

eg: $\begin{bmatrix} 2600\ ft^2 \\ \$300,000 \end{bmatrix}$ } 2 dimensional

A student wants to keep data of a house price & area

Physics Student	Mathematician	CS student
↗	$\vec{v}$	$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$
Direction & magnitude	He thinks a vector can be anything	

Episode 2:



$\hat{i}$ and $\hat{j}$ are the "basis vectors" of the xy co-ordinate system

Technical definition of basis:
The BASIS of a vector space is a set of linearly dependent vectors that span the full space.

Span:
The set of all possible vectors that you can reach with the linear combination of a given pair of vectors is called the span of these two vectors.

∴ The span of most pairs of 2-D vectors is all vectors of 2-D space.

SPAN in 2D
- Parallel vectors: Span of such vectors is a straight line (Linearly dependent)
- Non-Parallel vectors: Span of such vectors is the whole 2-D space (Linearly independent)

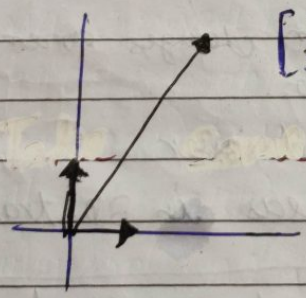
SPAN in 3D:
- 2 vectors: Trace out a plane
- 3-Vectors: (Third is span of first 2) Trace out same plane
- 3-Vectors: (3rd is different span) Trace whole 3-D space.

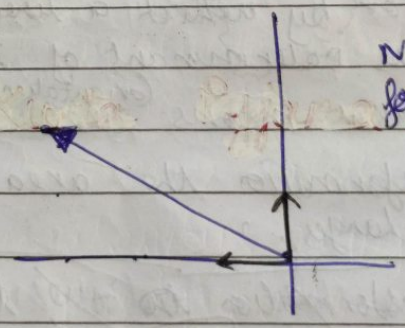
Episode 3:

Linear transformations: In general we should think of linear transformations as keeping grid lines parallel and evenly spaced.

- If lines, vectors get curved or transformed it is not linear transformation

eg. If we shift space by 90 degrees





New coordinates for $\hat{i}$ and $\hat{j}$ are

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

Therefore now the new coordinates of the vector is

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 0\times2 - 1\times3 \\ 1\times2 + 0\times3 \end{bmatrix} = \begin{bmatrix} -3 \\ 2 \end{bmatrix}$$

Thus ∴ We can now imagine multiplication of vectors.!!!

Episode 4:

Multiple Linear Transformation

eg. Composition of a rotation and a shear.

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}\left(\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{bmatrix} 1 & -1 \\ 2 & 0 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}$$

Shear · Rotation · Composition

← Read right to left

Always remember that multiplying 2 matrices like this has geometric meaning of applying one transformation then another.

Moreover, $M_1 M_2 \neq M_2 M_1$ where M_1, M_2 are matrices.

Also, $A(BC) = (AB)C$ ∴ Its associative

Episode 5: 3-D linear transformation:

Transformation is same only difference is that it happens in 3-D space. (All rules are the same).

3-D transformation is important for computer graphics & robotics.

Episode 6 DETERMINANT:

~~The area by which a~~ The factor by which a linear transformation changes area is called as Determinant of that transformation.

* -ve Determinant means area (orientation) has been flipped over

* In 2D transformation the area changes whereas in 3D transformation the Volume changes..

* In 3-D transformation ~~the~~ +ve determinant represent ~~that~~ the basis vectors are $\hat{i}\,\hat{j}\,\hat{k}$

* In 3-D transformation ~~the~~ determinant ~~is~~ use your right hand using pointing finger of $\hat{i}$, middle finger for $\hat{j}$ & thumb for $\hat{k}$ and if the transformation you can still use right hand & point $\hat{i}, \hat{j}, \hat{k}$ then the orientation has not changed and determinant is +ve. Now if you can do so at with left hand the determinant has been flipped and the determinant is -ve.

$\det\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right) = ad - bc$ Values of b,c determines how much the area has been squeezed.

Episode 7: $A^{-1}A$ = Identity matrix $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

Linear equations simply we need to find a vector x such that when it is taken through a transformation of A it lands on vector $\vec{v}$

$$A\vec{x} = \vec{v}$$

eg. $2x + 2y = -4$
$1x + 3y = -1$

$$\Rightarrow \underbrace{\begin{bmatrix} 2 & 2 \\ 1 & 3 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x \\ y \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} -4 \\ -1 \end{bmatrix}}_{v}$$

A is transformation
x is vector to be transformed.
v is required vector after transformation

When Determinant = 0. : The transformation in this case squishes the plane into a single line of a point in 2D and squishes the space onto a plane in 3-D.

RANK: It refers to the number of dimensions in the output of a transformation. i.e. if a transformation squished into a line then rank is 1. If a tranformation squishes into a plane the Rank is 2.

* The maximum rank of a matrix is its order. [Here we are talking only about square matrix]

→* COLUMN SPACE: It is the set of all possible outputs after a transformation eg. $A\vec{v} \longleftrightarrow$ "Column space" of A

* Eg 2: Span of columns of $\begin{bmatrix} 2 & -2 \\ 1 & -1 \end{bmatrix} \Longleftrightarrow$ Column space

Also, RANK is precisely the number of dimensions in the column space.

$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$ Zero vector is always in the column space

NULL SPACE (Kernel):

If a 2D transformation squishes in a 2D plane on a line then there are a whole bunch of vectors in opposite direction of the line that get squished onto the origin.

Similarly, If 3D space is squished onto a plane then there is a whole line of vectors that land on the origin and when 3D space is squished onto a line then there is a whole plane of vectors that land on the origin.

⇒ The set of vectors that land on the origin is called null space or kernels of the matrix.

→ Dot product: It is length of projection of one vector on the second vector multiplied by the length of second vector.

eg: let there be 2 vectors A & B.

$A.B = (\text{length of projection of B on A / A on B}) \cdot (\text{Length of A/B})$

-We dot product means vectors are pointing in opposite direction.

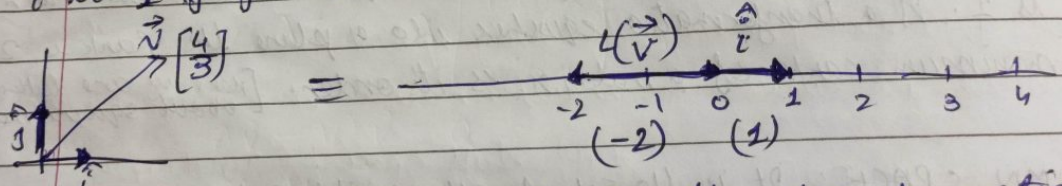
A dot product makes one of the vectors as transformation from 2D space to 1D space.

eg: $\begin{bmatrix} x \\ y \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} = x.A + y.B$

→ Cross product: It is area of ||gm between the vectors.
i.e. determent of matrix made by 2 vectors.

→ DUALITY:

eg. let thing of vector $\begin{bmatrix} 4 \\ 3 \end{bmatrix}$ which takes $\hat{i}$ to 1 & $\hat{j}$ to −2



A consequence of linearity is that after the transformation, the vector will be 4 times where the $\hat{i}$ lands i.e. 1 plus 3 times the place where $\hat{j}$ lands, which in this case implies that it lands on −2.

When we do this calculation purely numerically, it is matrix vector multiplication.

$$\begin{bmatrix} 1 & -2 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix} = 4.1 + 3(-2) = -2$$

∴ The idea of duality is that anytime we have a linear transformation from some space to the number line, it is associated with a unique vector in that space, in the sense that performing the linear transformation

is the same as taking a dot product with that vector

* Numerically this is one of those transformations which is described by a matrix with just one row, where each column tells you the number that each basis vector lands on.

CHANGE OF BASIS:

Suppose we have our basis vectors and jennifer has her own basis vectors and for our basis vectors we are referring to some vectors and if change of basis transformations helps us to find the vector in terms of jennifers basis vectors.

eg. let there be a transformation matrix say $\begin{bmatrix} 1 & 2 \\ 2 & -1 \end{bmatrix} = A$

The basis vectors i and j are therefore taken to $\begin{pmatrix} 1 \\ 2 \end{pmatrix}$ and $\begin{pmatrix} 2 \\ -1 \end{pmatrix}$

Let say there is some Transformation $T = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$

$\begin{bmatrix} 1 & 2 \\ 2 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$ ⟹ This will give vector in normal coordinates.

A V

Let V be in the new co-ordinate system and T be the transformation.

$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 2 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$ → This gives the transformed vector in normal coordinate

T A V

$\begin{bmatrix} 1 & 2 \\ 2 & -1 \end{bmatrix}^{-1} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 2 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$ → This gives transformed vector in new co-ordinate system

A^{-1} T A V

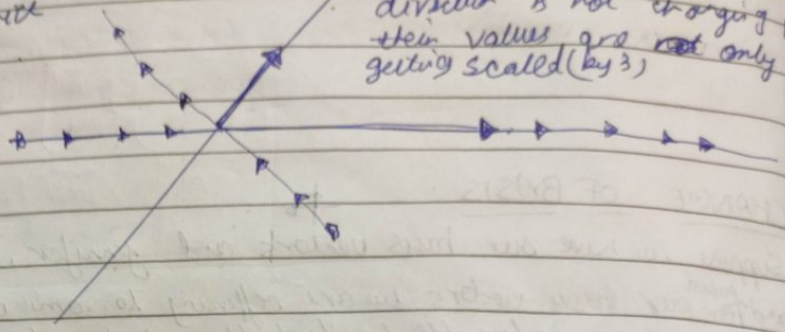
General formula for transformation in new coordinates: $(A^{-1}TA)V$

→ Eigen Vectors and Eigen values:

Eigen vectors are some special vectors whose direction i.e. span does not change after a particular linear transformation. In 3D. Space Eigen vector represent the axis of rotation as no change is observed after transformation.

eg. let there be transformation of the form $\begin{pmatrix} 3 & 1 \\ 0 & 2 \end{pmatrix}$

Vectors lying on $\begin{pmatrix} -1 \\ 1 \end{pmatrix}$ Are also a Eigen vectors which get scaled by a factor of 2.



Here vectors on the x axis are Eigenvectors as their direction is not changing but their values are not only getting scaled (by 3)

∴ we can Write Eigen vectors as

$A\vec{V} = \lambda\vec{V}$ A is transformation, V is Eigenvector.

Eigenvalue is the factor by which an Eigen vector is scaled in a linear transformation. In this case λ.

$A\vec{V} = (\lambda I)\vec{V}$

Here λI is $\lambda \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

$A\vec{V} - (\lambda I)\vec{V} = 0$

$(A - \lambda I)\vec{V} = 0$

$(A - \lambda I) = 0$ This is true when V is a zero vector also when the determinant $(A - \lambda I)$ is zero This mean $(A - \lambda I)$ is a transformation that squishes a plane onto a line.

So by equating det $(A - \lambda I)$ to 0 we get the value of λ, which is the Eigen value.

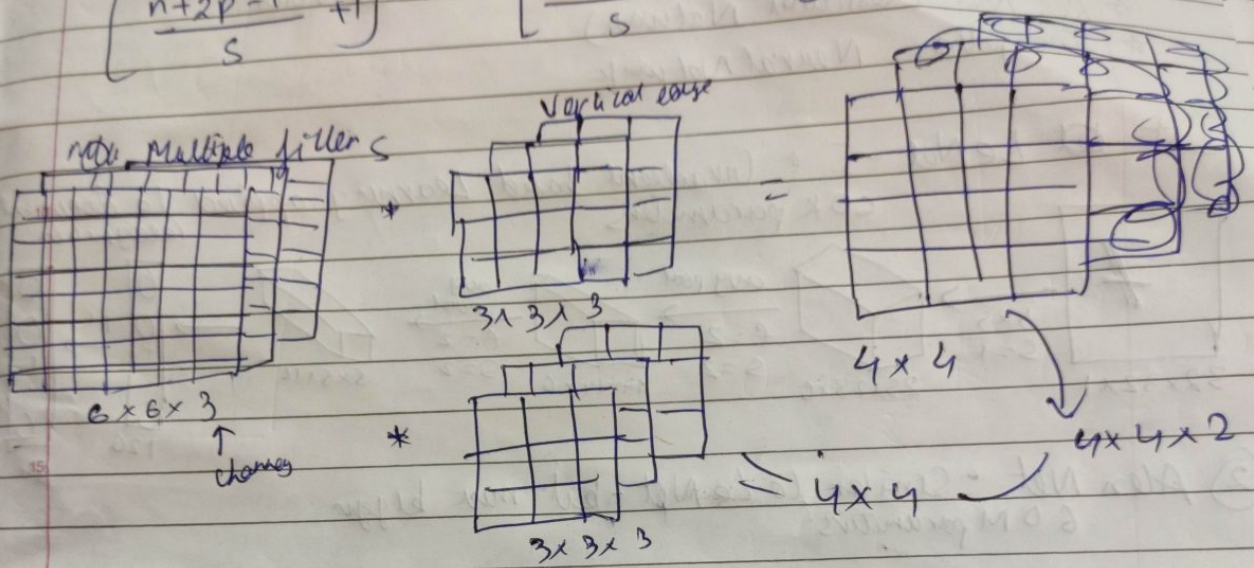
EIGEN BASIS:

Choosing our basis vectors as Eigen vectors classifies them as Eigenbasis

- This is helpful in multiplying matrices multiple times.
- If we are given a non-diagonal matrix that are actually the Eigenvectors then by transforming them back and reducing them to a diagonal form, makes it easier to multiply them since in a diagonal matrix the diagonals are simply raised to power of how many times they want to be multiplied and then multiplying that matrix with inverse of the transformation to get the desired output

Course 4
Week 1: Padding
Convolution formulas
$n \times n$ image
padding p
$f \times f$ filter
strides

$$\left[\frac{n+2p-f}{s}+1\right] \times \left[\frac{n+2p-f}{s}+1\right] \text{ matrix}$$



Summary: $n \times n \times n_c$ $\quad * \quad f \times f \times n_c \quad \rightarrow \quad \frac{n-f+1}{4} \times \frac{n-f+1}{4} \times n_c'$

eg. $6 \times 6 \times 3 \qquad 3 \times 3 \times 3$

n_c' ← number of filters used

Summary of notation

If layer l is a convolution layer.

$f^{[l]}$ = filter size
$p^{[l]}$ = padding
$s^{[l]}$ = stride
$n_c^{[l]}$ = number of filters

Input: $n_H^{[l-1]} \times n_W^{[l-1]} \times n_c^{[l-1]}$
Output: $n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$

$$n_{H/W}^{[l]} = \left[\frac{n_H^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1\right]$$

Each filter is: $f^{[l]} \times f^{[l]} \times n_c^{[l-1]}$

Activations: $a^{[l]} \rightarrow n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$

$A^{[l]} = m \times n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$

Weights: $f^{[l]} \times f^{[l]} \times n_c^{[l-1]} \times n_c^{[l]}$
↖ filters ↑ layer l

Bias $n_c^{[l]} - (1, 1, 1, n_c^{[l]})$

7:52 pm. 29-09-2021

DEEP LEARNING SPECIALIZATION

Courses in this specialization

* Neural Networks & Deep learning
* Improving Deep Neural Networks: Hyper parameter tuning, Regularization and optimization
* structuring your M.L. project
* Convolutional Neural Network
* Natural Language Processing: Building sequence models.

Supervised learning

Input (x)	Output (y)	Application	
Home features	Price	Real Estate	Standard NN
Ad, User info	Click on ad?	Online Adv.	
Image	object (1, ... 1000)	Photo tagging	CNN
Audio	Text transcript	Speech recog	Recurrent Neural N.
English	Chinese	Machine Translation	
Image, Radio translation	Posito of other cars	Autonomous driving	

Week 2: Binary classification: In this our aim is to learn a classifier that the input an image represented by vector X and predict corresponding label y is 1 or 0. eg. if the image is of a cat, y will be 1 or else it will be 0.

~~eg. all image of~~ * logistic regression is an algorithm for binary classi...

NOTATION:

① A Training eg. is denoted by a pair (x, y) where, $x \in R^{n_x}$ (n_x-dimensional feature vector)

$y \in \{0, 1\}$ (label)

② m : (no. of training eg), $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots (x^{(m)}, y^{(m)})\}$

m_{train} : to emphasize the no. of training eg.

m_{test} : testing examples

$$X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}_{n_x}$$

$X \in R^{n_x \times m}$

$X.shape = (n_x, m)$

$Y = [y^{(1)} \; y^{(2)} \; \cdots \; y^{(m)}]$

$Y \in R^{1 \times m}$

$Y.shape = (1, m)$

Logistic Regression: It is used when dependent variable is categorical
eg. To predict whether an email is spam (1) or not (0)

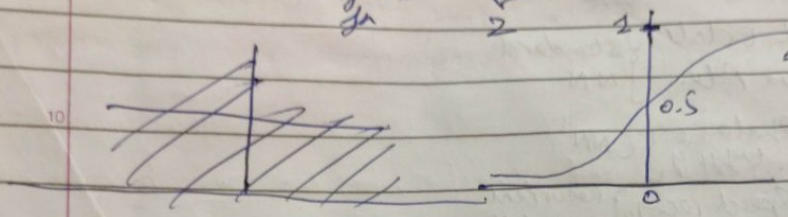
Given x, we want $\hat{y} = P(y=1/x)$

$\hat{y}$ ⇒ estimate of y, probability of the chance that y is equal to one, given the input features x.

$x \in R^{n_x}$

Parameters : $w \in R^{n_x}$, $b \in R$

Output $\hat{y} = \underset{\text{Sigmoid fn}}{\sigma}(w^T x + b)$ (w^T ⇒ transpose of matrix w)



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

If z large (+ve)
$\sigma(z) \approx \frac{1}{1+0} = 1$

If z large (-ve)
$\sigma(z) \approx \frac{1}{1+\infty} \approx 0$

To train the parameters we need to define a cost fun

$\hat{y}^{(i)} = \sigma(w^T x^{(i)} + b)$, where $\sigma(z^{(i)}) = \frac{1}{1+e^{-(i)}}$

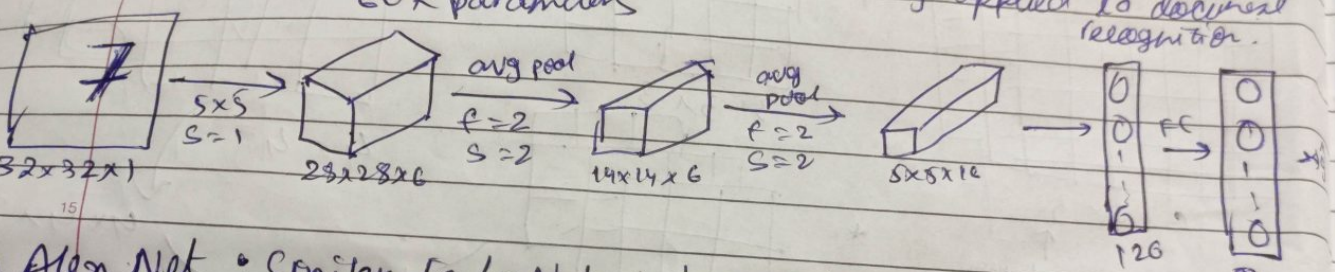
Given $(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$, want $\hat{y}^{(i)} \approx y^{(i)}$

Types of layer in a convolutional network
- Convolution - Pooling - Fully connected

★ Why Convolutions ① Parameter sharing: A feature detector (eg. vertical edge detection)
② Sparsity of connection.

★ Few Classic networks
① LeNet-5 ② Alex Net ③ VGGs
★ Res Net (Residual Network)
★ Inception Neural Network

① LeNet-5 : Gradient based Learning applied to document recognition.
60K parameters



② Alex Net : Similar to LeNet - but much bigger
60M parameters

③ VGG-16 : CONV = 3×3 filter, S=1, same
MAX POOL = 2×2, S=2
} It is relatively deep network.
eg: 224×224×64

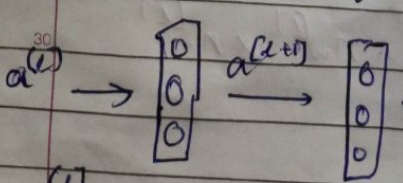
224×224×3 → CONV 64 ×2 → 224×224×64 → POOL → 112×112×64 → CONV 128 ×2 → 112×112×128 → POOL

56×56×128 → CONV 256 ×3 → 56×56×256 → POOL → 28×28×256 → CONV 512 ×3 → 28×28×512 → POOL

→ CONV 512 ×3 → 14×14×512 → POOL → 7×7×512 → FC 4096 → FC 4096 → Softmax 1000

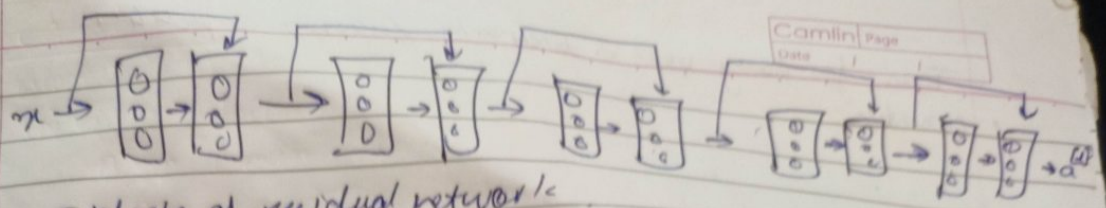
In this we are doubling CONV layer everytime and while pooling we ~~reduce~~ reduce matrix are reduced

★ Res Net : Residual Block

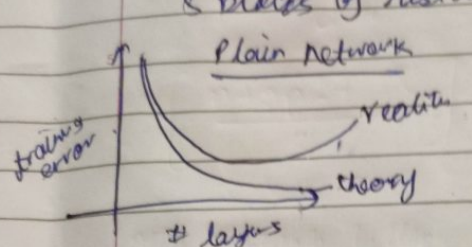


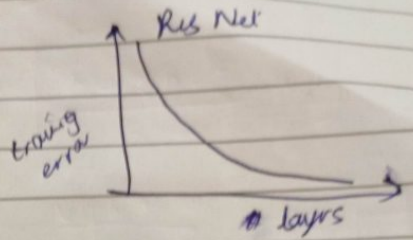
$a^{[l]} \rightarrow$ linear $\rightarrow$ Relu $\rightarrow a^{[l+1]} \rightarrow$ linear $\rightarrow$ Relu $\rightarrow a^{[l+2]}$

$z^{[l+1]} = W^{[l+1]} a^{[l]} + b^{[l+1]}$



5 blocks of residual network





☆ **Mobile Net** • Low computation cost at deployment useful for mobile & embedded vision applications

Normal vs depth wise – Seperable convolutions

* Normal convolution

$n \times n \times n_c \cdot f \times f \times n_c = n_{out} \times n_{out}$

eg. $6 \times 6 \times 3 \quad 3 \times 3 \times 3 = 4 \times 4$

Computational cost = # filter params × # filter positions × # of filters

Depthwise Convolution

$n \times n \times n_c$	$f \times f \times n_c$	$n_{out} \times n_{out} \times n_c$
$6 \times 6 \times 3$	$3 \times 3 \times 3$	$4 \times 4 \times 3$

Computational c = # filter params × # filter positions × # of filters

$3 \times 3 \times 4 \times 4 \times 3$