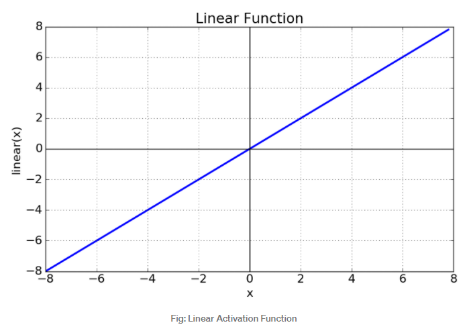
# Neuralt netværk

## Aktiveringsfunktion

<https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6?gi=56bd74d85bcf>

Ideen med en aktiveringsfunktion er, at ud fra den værdi, som et neuron giver, så beslutter aktiveringsfunktionen, om den værdi skal med videre i netværket. Der findes flere typer af aktiveringsfunktioner, heraf både lineære og u lineære. Fremover skrives aktiveringsfunktion som AF.

**Lineær aktiveringsfunktion**

Her ses en lineær AF.

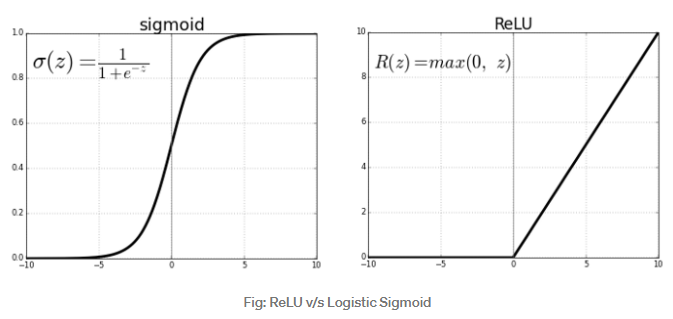
Formel:

Denne AF hjælper ikke på at mindske kompleksiteten

**Nonlineær aktiveringsfunktion**

Den mest benyttet AF. Man ser på om funktionen er differentiabel (finder en hældning) og monoton.

Mindsker kompleksiteten. Ulempen ved en nonlineær AF er, at den smider data væk, hvis dataen ligger udenfor dens rækkevidde.



## Gradient descent

<https://www.youtube.com/watch?v=sDv4f4s2SB8>

Det er en proces, der sker i backpropagation-delen.

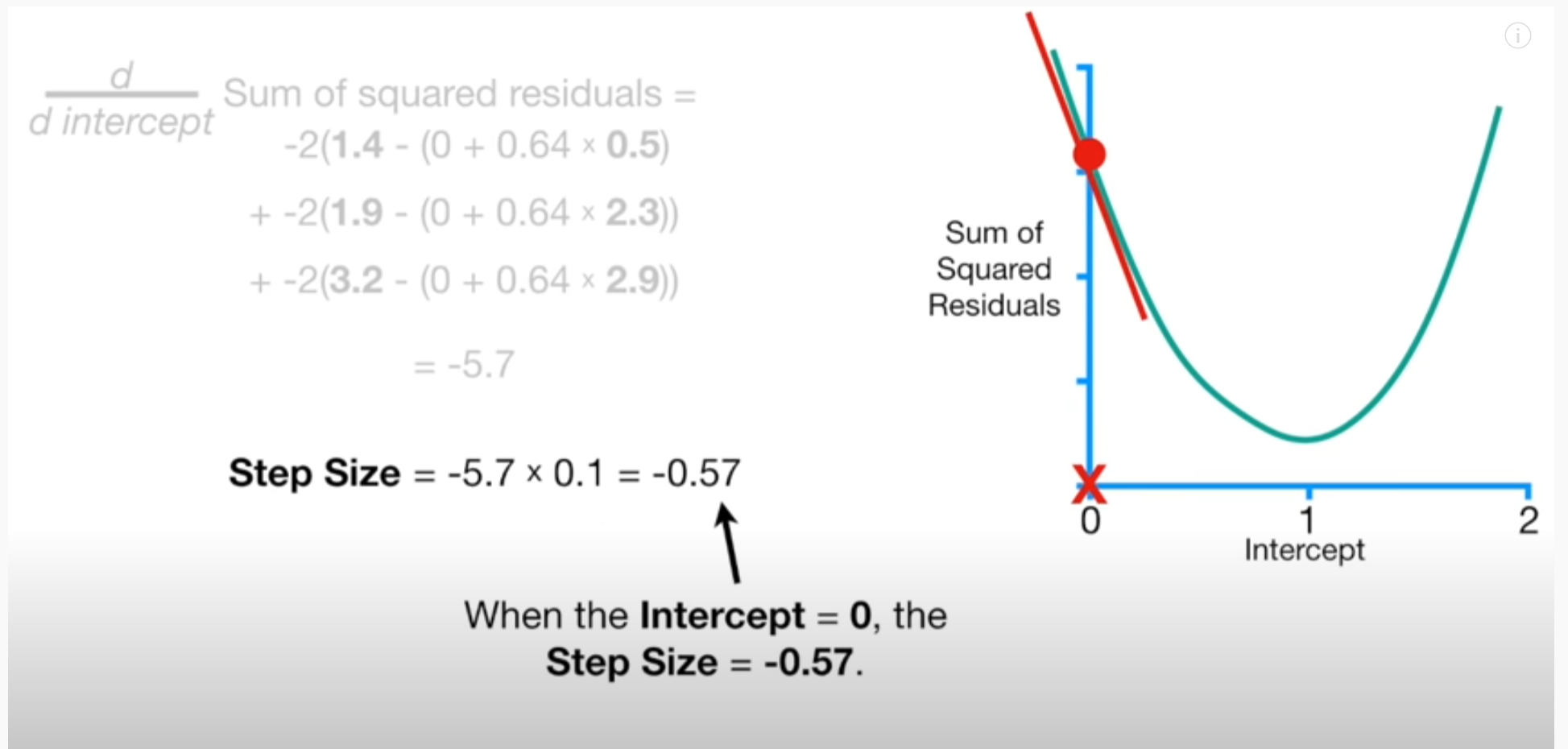
Exploding and vanishing gradients – Den eksploderer hvis normalen af vægtmatricen er over 1 og forsvinder, hvis den er under 1.

Når den optimale linje skal findes for sine punkter, benyttes gradient descent. Gradient descent sørger for at, hvis der er stor afstand med en loss funktion, så tager den store skridt, og størrelsen reduceres i takt med at den kommer tættere på den optimale løsning.

Opsummeret, så sørger gradient for at finde den bedste tilpassende linje / flade til alle datapunkterne på baggrund af loss funktionens hældning og skæring (altså hvor den møder et toppunkt på kurven).

## Læringskurve / learning rate

Gradient descent bestemmer step size sammen med læringskurven. Når gradient descent differentieres, så finder man den hældning, som kurven har i det pågældende punkt. Den hældning ganges med learning rate, og derved bestemmes step size på den måde for at undgå at man tager et for stort skridt, der resulterer i at man overskrider den laveste afvigelse af den fittede linje. Gradient descent stopper, når step size er mindre end learning rate værdi.



## Batch Normalisering

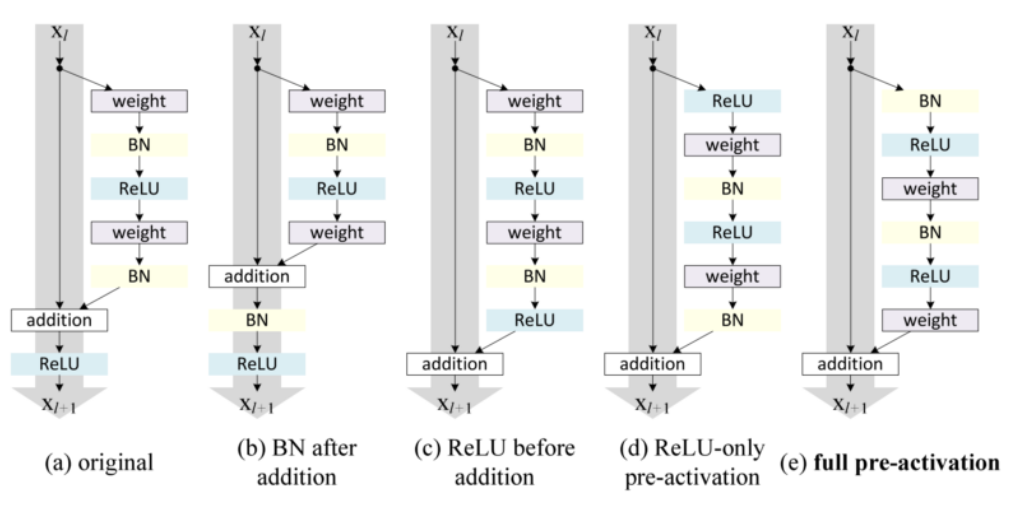
Batch normalization kan tilføjes til konvolutional og fully connected lag, og kan hjælpe til at forbedre gradiant flow, som også kan beskrives som det generelle forløb af netværkets optimering i træningen. Det kan derfor betyde at der kan bruges højere learning rate, da det tillader en mere aggressiv optimering af netværket. Batch normalization virker ved at normalisere outputtet af hvert activation map hvis det bliver brugt på et convolutional lag, eller generelt på hele aktiveringen for et fully connected lag. I normaliseringen bliver alle værdier tvunget til at være normalfordelt, og bliver derfor rettet ind. Det sker ved at hver værdi trækkes fra midedelværdien af alle værdier og derefter divideres med kvadratroden af variansen. Der kan også trænes to værdier, shift og scale, til at blive brugt som faktorer i udregningen til at rette ind på værdierne hvis mean er langt fra 0 eller variansen er langt fra 1.

## Residual Blocks

Residual blocks anvendes, fordi meget dybe netværk er svære at træne pga. forsvindende eller eksploderende gradienter. De tager inputtet og føder det fremad, hvor det adderes efter en standard eller depthwise separable convolution lige før den tilhørende aktiveringsfunktion.

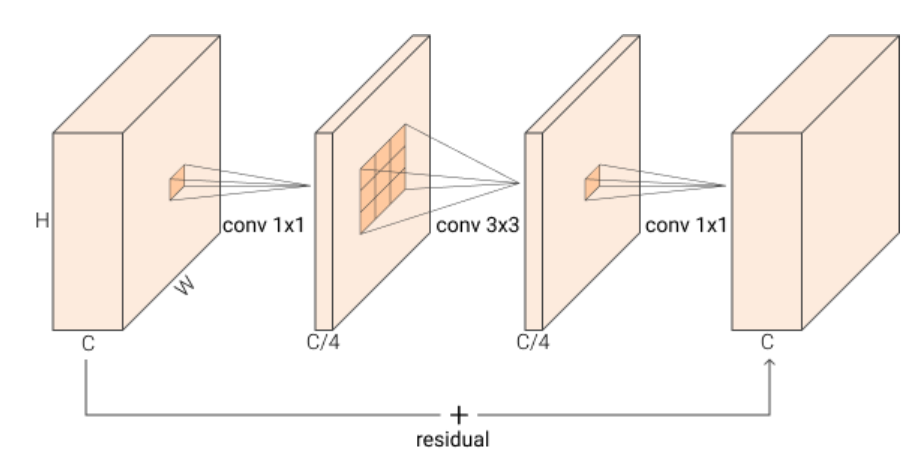
Der findes forskellige typer af residuals. Herunder ses forskellige metoder, hvorpå man kan behandle de data, som trækkes ud og sendes videre i netværket.

Jeg har set billedet omvendt flere gange. En residual er i bund og grund bare en skip connection, hvor det data, som kommer som input bare adderes til outputtet af de blokke imellem.

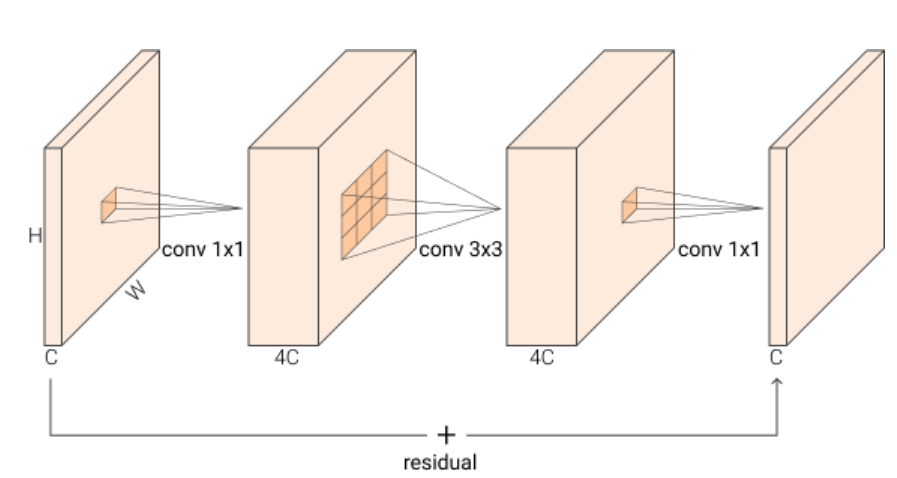


## Inverted Residual blocks

En almindelig residual blok ser sådan ud.



En inverted residual blok ser sådan ud.



Og kaldt en omvendt bottleneck blok.

<https://paperswithcode.com/method/inverted-residual-block>

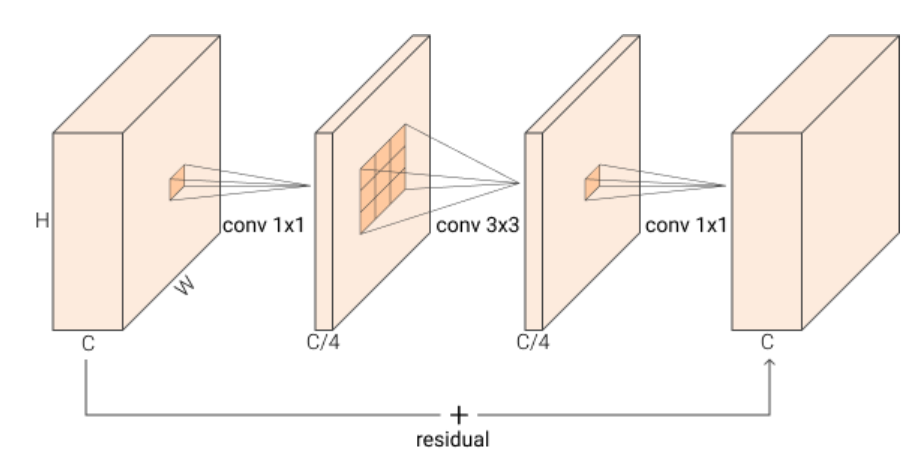
Fordelen er man reducerer antallet af parametre betydeligt.

## Bottlenecks

<https://towardsdatascience.com/residual-bottleneck-inverted-residual-linear-bottleneck-mbconv-explained-89d7b7e7c6bc>

Der er måske en misforståelse af bottlenecks. Beskrivelsen fra rapporten skriver, at det er fordi at aktiveringsfunktionen er lineær. Efter at have læst det igen er jeg kommet frem til at en bottleneck formegentlig er den 1x1xdybde konvolution, der er i netværket. Dvs. når man konverterer dybdelagene til et enkelt lag. Samtidig med når man konveterer tilbage fra 1 lag til samme størrelse dybdelag, som før.

Lineær bottleneck er så en bottleneck block, som ses her, men uden en aktiveringsfunktion.



Beskrivelsen fra opgaven:

*Linear bottleneck betyder at det sidste lag af en residual blok fjernes aktiveringsfunktionen, som giver et lineært output i stedet for ReLU's ulineære output. Det giver et mindre tab af data på bekostning af ydeevne, og derfor kaldes det for bottlenecks. Sammen med inverted residual blokke, så gør det databehandlingen mere hukommelseseffektiv. hvor billede viser MobileNet V1's lagstruktur og viser MobileNet V2's lagstruktur. Forskellen ses på lag, hvor der kun er et enkelt stride, så anvendes inverted residual blocks og lineære bottlenecks, mens hvis der en stride på mere end en, så anvendes kun den lineære bottleneck.*

# Regulering

## State Space

Det er en lineær repræsentation af et dynamisk system i enten kontinuert eller diskret form.

Ud fra state space kan man finde systemets stabilitet, hvilket er egenværdien af matrix A. Mindre end 0 giver et stabilt system

State space sørger for at man kan konvertere bevægelsesligninger for en mekanisk model om til matricer og vektorer. State beskriver en gruppe variabler, som beskriver historien af et system til at forudsige fremtidigt variabler. Antallet af state variabler skal være lig med antal af storage elementer i systemet.