# Ansiktsuttrycksklassificerare

## Introduktion

### - Bakgrund

Deep Learning eller neurala nätverk är en del av Maskininärningen där man använder sig av neuroner precis som i människans hjärna där dem skapar kopplingen mellan varandra och lär sig från föregående neuroner. Deep Learning är särskilt bra att hitta struktur i data med komplexa och omfattande datamängder och lösa komplicerade problem. De problem som DL används oftast för är bildigenkänning, språkbehandling, röstigenkänning, självkörande fordon och Chat GPT.

### - Syfte och Frågeställning

Syftet med denna uppgift är att träna olika modeller och välja den bästa modellen som överfittar eller underfittar valideringsdata minst. Målet för detta klassificeringsproblem är att hitta en modell som kan korrekt identifiera rätt ansiktsuttryck. Resultatet mäts med accuracy och visualiseras med plottar som visar korrelationer mellan träning loss och validation loss samt accuracy och validation accuracy. Denna visualisering är enkel och passar bra för detta klassificeringsproblem för att man vill om validation data överfittar eller underfittar modellen.

## Databeskrivning - EDA

Datasetet som kommer att användas i modelleringen är ”Face expression recognition dataset” från Kaggle som innehåller 2 mappar där den ena är tränings set med ca 28 000 bilder och den andra validerings det med ca 7000 bilder. Bilderna är uppdelade i klasser: angry, disqust, fear, happy, neutral, sad och surprise.
Bilderna ska laddas upp med 48 x 48 storlek med 3 färger per bild. (48, 48, 3). Därefter kommer de preprocesseras med ImageDataGenerator() och delas i träningsdata och validationdata för att underlätta processen.

## Metod och modeller

**Modell 1** är ett exempel kod klistrat in för att förstå tanken bakom koden så jag kommer inte fördjupa mig i den.

**Modell 2** bestämde jag att börja bygga med Keras Tuner för att se hur den bästa modellen kommer performa med validation data. Den tunade många hyperparametrar. Jag började med att introducera **Early Stopping** vilket borde inte ha varit med. Därefter kommer loopade

modellen igenom 5 **convolutional** layers med början på 64 neuroner som stegvis höjdes med 128 neuroner till en maxantal på 258 neuroner. Näst i loopen var olika **kernel size**, **activation** funktion, **input layer**, och **padding**. Efter varje loop optimeras layern men Batch Normalization, får en bestämd pooling och en Droput som ändrade värden varje gång för att hitta den optimala värden.
Efter alla convolutional layers kommer en **flatten** layer. Därefter en till loop som reglerade antal **Dense** layers med olika värden. Efter varje loop tilldelas det Batch Normalization standardiserings funktion samt med regulatorn Droput som hade olika värden. Sist kommer **output** Dense layer med 'softmax' activation eftersom vi har 7 klasser vilket är 7 möjliga outputs därför sätter vi "categorical crossentropy" som loss funktion. Modellen kompileras och **Learning rate** justeras. Resultatet mäts med accuracy.

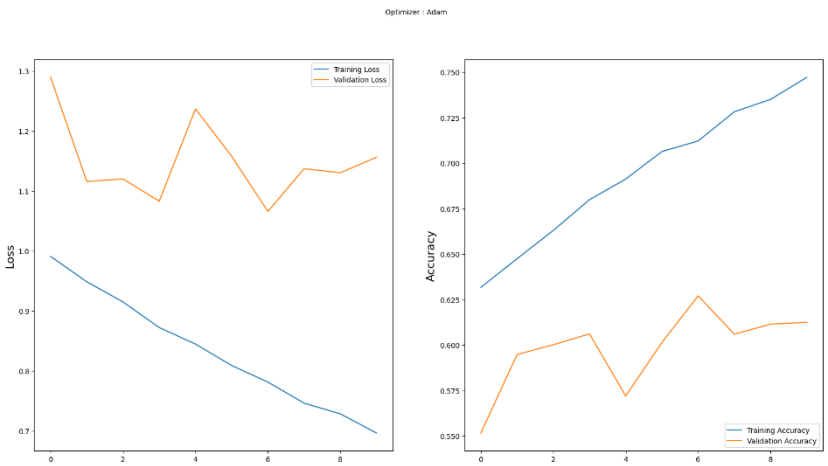
Regulariseringen kom i nästa ruta där jag skapade en lista med "**callbacks"** som innehhåller Early Stopping, Reducering av Learning Rate och Checkpoint ifall träningen crashar. Learning rate ska börja med 0.009 och multipliceras med 0.99 så att den går från 0,009 till 0.0089 till 0.0087 osv.

**Modell 3** har skapats med ny Keras Tuner Klass. Svaret på varför kommer under analysen. Modellen var mindre komplex för att snabba upp tuningen Det finns några ändringar med antal convolutional layers och antal neuroner u första loopen. Det blir flera layers och högre maxantal neuroner. Andra loopen har färre **Dense layers**. Ingen Early stopping ska vara med i tuningen för att snabba upp processen. Den kommer i andra rutan med andra regulators.

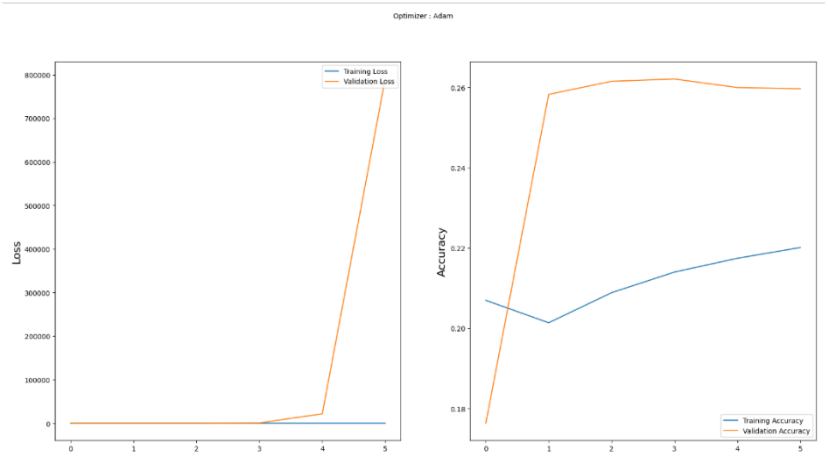
**Modell 4** var baserad på den bästa modellen från andra Keras tuner. Där tog jag bort en convolutional layer för att se om den performar bättre och ändrade antal neuroner i till pyramid formar. Reduceringen av learning rate var justerat enligt den bästa modellen från Keras Tuner. Sist var det "pool size" på sista conv. layer som behövde justeras för att få träningen i gång.

## Projekt, Resultat och Analys

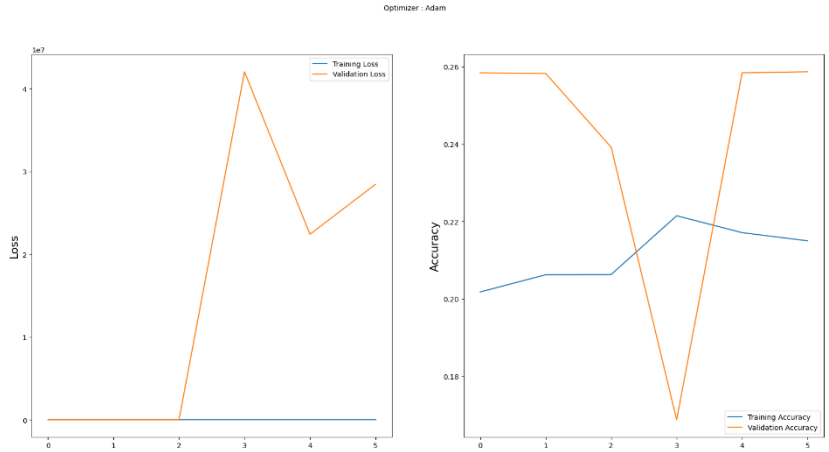
Modell 2 - Den bästa tunade modellen enligt Keras Tuner överfittar valideringsdata.



Modell 3 var – Den bästa enligt Keras Tuner2




Modell 4 som jag skapade genom att göra några ändringar på modell 3




## Slutsats och förslag på potentiell vidareutveckling

För bättre resultat skulle man kunna tuna hyperparametrarna ännu mer för att hitta den perfekta antal layers och antal neuroner. Detta kunde göras med mindre dataset för att snabba upp processen. I stället för att använda processorn för att modellera data kunde man använda grafikkortet vilket visade sig snabbare enligt mina klasskamrater. Flera epoker betyder flera kombinationer och potentiellt bättre resultat. Det finns några andra regulariseringstekniker som hade kunnat ge bättre resultat.

Ett bra förslag på en vidareutveckling är att träna ett till dataset och få ihop de så att bildklassificerare kan igenkänna flera objekt på bilden.

Min plan från början var att skapa ett litet program för barn som vill bli skådespelare eller gå på teater. Användare ska få läsa en kort text/situation och få reagera på den. Korrekt reaktion blir godkänt och då går man till nästa text.