Humörigenkänning från Röst med Spotify-integration



Ming Fondberg

EC Utbildning

Grupp projet

2024-10

# Abstract

Detta projekt utforskar användningen av Deep Learning (DL) för att analysera röstinspelningar och förutsäga användarens humör, vilket sedan kopplas till personliga musikrekommendationer via Spotify. Systemet är implementerat med FastAPI som en REST API-tjänst och en Tkinter-baserad applikation för att spela in ljud och visa humörprediktioner. En förtränad konvolutionell neuralt nätverk (CNN) används för att klassificera ljud i tre humörkategorier: lugn, energisk och glad. Projektet visar en fungerande prototyp och diskuterar framtida möjligheter för att förbättra användarupplevelsen genom fullständig automatisering av Spotify-integratione.

**Nyckelord**: Deep Learning, Humörigenkänning, Ljudanalys, Webbapplikation, REST API, FastAPI, Spotify-integration.

Innehållsförteckning

[1 Inledning 1](#_Toc176961624)

[1.1 Bakgrund 1](#_Toc176961625)

[1.2 Problemformulering 1](#_Toc176961625)

[1.3 Syfte 1](#_Toc176961625)

[1.4 Frågestställningar 2](#_Toc176961625)

[1.5 Grupparbete 2](#_Toc176961625)

[2 Teknisk bakgrund](#_Toc176961626) 3

[2.1 Depp Learning för ljudanalys 3](#_Toc176961627)

[2.1.1 CNN 3](#_Toc176961628)

[2.1.2 MFCC 3](#_Toc176961629)

[2.2 FastAPI 4](#_Toc176961631)

[2.3 Applikation 5](#_Toc176961631)

[2.4 Spotify-integration](#_Toc176961631) 6

[3 Metod 7](#_Toc176961632)

[3.1 Systemarkitektur 7](#_Toc176961633)

[3.2 Test och utvärdering 7](#_Toc176961631)

[4 Resultat 8](#_Toc176961635)

[5 Diskussion 9](#_Toc176961635)

[5.1 Integrationsutmaningar 9](#_Toc176961627)

[5.2 Framtida utveckling 10](#_Toc176961627)

[6 Slutsatser 11](#_Toc176961636)

[7 Självutvärdring 12](#_Toc176961637)

[Källförteckning 13](#_Toc176961639)

# Inledning

## Bakgrund

Deep Learning (DL) har under de senaste åren blivit en central och snabbväxande teknologi inom multimediaapplikationer, såsom röstigenkänning, emotionell analys och rekommendationssystem. DL möjliggör automatiserade analyser som kan utföras både snabbt och effektivt, vilket revolutionerar sättet vi interagerar med teknologi. I detta projekt utforskas hur DL kan användas för att analysera röstinspelningar och klassificera användarens känslomässiga tillstånd i realtid. Genom att använda en DL-modell kan applikationen identifiera olika humör som exempelvis lugn, energisk eller glad, baserat på specifika ljudegenskaper i rösten.

Projektet syftar till att skapa en innovativ och användarvänlig lösning där denna information används för att tillhandahålla personliga musikrekommendationer via Spotify. Denna lösning, baserad på en FastAPI-tjänst i kombination med en desktop-applikation byggd med Tkinter, erbjuder möjligheten att analysera inspelade ljudklipp och direkt förse användaren med en spellista som matchar deras nuvarande humör. Genom att integrera DL-modellen i applikationen kan systemet ge en automatiserad och dynamisk rekommendation, som gör musikupplevelsen mer anpassad till användarens emotionella tillstånd.

## Problemformulering

Att analysera känslor från röstinspelningar på ett manuellt sätt är både tidskrävande och subjektivt, vilket innebär att det kan vara svårt att dra konsekventa och snabba slutsatser om en persons emotionella tillstånd. Med dagens teknik finns det potential att använda DL för att göra denna process mer exakt och effektiv. Målet med detta projekt är att automatisera känsloigenkänningen genom en DL-modell och att koppla resultaten direkt till Spotify för att ge musikrekommendationer som speglar användarens känslor. Detta kräver att man skapar en smidig integration mellan DL-modellen, användargränssnittet och Spotify, samt en effektiv kommunikation mellan olika systemkomponenter för att säkerställa realtidsanalys och automatiserad rekommendation. En av de främsta utmaningarna i projektet är att hantera denna integration, speciellt vad gäller automatisering av spellistrekommendationer och -uppspelning, som är central för att uppnå en sömlös användarupplevelse.

## Syfte

Det övergripande syftet med projektet är att skapa en funktionell prototyp som visar hur DL kan användas för att förutse humör baserat på röstinspelningar och därefter ge musikrekommendationer via Spotify. Specifika mål inkluderar:

* **Implementera en DL-modell**: En modell baserad på CNN ska utvecklas för att analysera ljud och klassificera känslor som lugn, energisk eller glad.
* **Integrera DL-modellen i en applikation**: Denna applikation, byggd med Tkinter, ska låta användaren spela in sitt röstljud, analysera detta ljud och därefter visa en känsloprognos.
* **Skapa en REST API-tjänst med FastAPI**: API

möjliggör kommunikation mellan applikationen och DL-modellen och säkerställer att ljudanalysen kan genomföras och resultat returneras till applikationen.

* **Koppla känslorna till Spotify-spellistor**: Applikationen ska rekommendera spellistor som överensstämmer med det förutspådda humöret för att göra musikupplevelsen mer personlig och stämningsanpassad.
* **Identifiera och hantera tekniska utmaningar**: Arbeta med och lösa problem relaterade till integration och kommunikation mellan komponenterna, samt förbättra modellens prestanda och noggrannhet för att ge användaren en konsekvent och effektiv upplevelse.

## Frågeställningar

* Hur kan DL effektivt integreras i en desktop-applikation för realtidsanalys av ljud?
* Vilka tekniska krav behövs för att säkerställa att ljudinspelningar kan analyseras och klassificeras med hög precision i realtid, och hur kan DL-modellen optimeras för att fungera effektivt i en desktop-miljö?
* Vilka utmaningar uppstår vid länkning av humörigenkänning med en extern tjänst som Spotify?
* Hur kan kommunikation mellan FastAPI och Spotify effektiviseras för att säkerställa snabb rekommendation och uppspelning av spellistor som speglar användarens känslor?
* Hur kan systemet utökas för att förbättra användarupplevelsen?

## Grupparbete

Projektarbetet delades upp i tre delar där vi alla hade sitt egna ansvarsområde,

David arbetade med Applikationen och samarbetade med integrationen.

Muhannad arbetade med pipelinen och samarbetade med integrationen.

Parsan arbetade med modellen och datasett samt samarbetade med integrationen.

Ming arbetade med applikationen och modellen samt samarbetade med integrationen.

# Teknisk bakgrund

## Deep Learning för ljudanalys

## Deep Learning (DL) har visat sig vara mycket effektivt för analys av ljuddata. Inom DL är konvolutionella neurala nätverk (CNN) och rekurrenta neurala nätverk (RNN) två av de mest använda teknikerna, särskilt för bild- och ljudklassificering. Genom att använda CNN kan specifika mönster i ljuddata identifieras och klassificeras. I detta projekt är CNN-modellen tränad för att klassificera ljud i tre olika känslotillstånd: lugn, energisk och glad. Modellen använder sig av en teknik kallad Mel-frekvenscepstrumkoefficienter (MFCC), som omvandlar ljuddata till en mer hanterbar form för analys och möjliggör klassificering baserad på dessa egenskaper.

#### 

### CNN

Konvolutionella neurala nätverk (CNN) är en typ av artificiellt neuralt nätverk, framtaget för att analysera bild- och ljuddata. CNN fungerar genom att extrahera funktioner från inmatade data och identifiera specifika mönster som används för klassificering. Dessa nätverk använder sig av en serie lager, där konvolutionslager är centrala för att extrahera viktiga funktioner från data. Ett konvolutionslager består av en uppsättning filter, eller kernlar, som appliceras på data för att generera aktiveringskartor (feature maps). Genom att analysera aktiveringskartorna identifierar nätverket mönster som är relevanta för specifika klasser.

I projektet tränades CNN-modellen för att upptäcka mönster relaterade till känslotillstånd i ljuddata. Dessa mönster är subtila och kan omfatta tonhöjd, intensitet och variation i ljud, som är avgörande för att klassificera känslor som lugn, energisk eller glad. CNN är särskilt lämpligt för denna typ av analys eftersom modellen kan fånga spatiala beroenden i ljuddata, vilket gör det möjligt för modellen att skapa en generaliserbar representation av känslomässiga mönster.

### MFCC

Mel-frekvenscepstrumkoefficienter (MFCC) är en metod för att extrahera funktioner från ljudsignaler, som är vanligt förekommande i röst- och ljudigenkänning. MFCC omvandlar tidsdomänsignalen till frekvensdomän, vilket gör det möjligt att analysera ljudets spektrala innehåll. Genom att använda en logaritmisk skalning och omvandla data till Mel-skalan, som bättre efterliknar hur människor uppfattar ljud, blir det lättare att identifiera skillnader mellan olika känslor i ljudet.

MFCC ger en representation som är både komprimerad och mer robust mot brus och variationer, vilket förbättrar modellens prestanda och noggrannhet vid känsloigenkänning.

Python

def predict\_mood(audio\_data):

mfcc = librosa.feature.mfcc(y=audio\_data, sr=22050, n\_mfcc=13)

mfcc = np.pad(mfcc, pad\_width=((0, 0), (0, 100 - mfcc.shape[1])), mode='constant')

mfcc = mfcc.T # Transponera för korrekt form

mfcc\_scaled = np.expand\_dims(mfcc, axis=0)

prediction = model.predict(mfcc\_scaled)

mood\_index = np.argmax(prediction)

mood\_map = {0: "calm", 1: "energetic", 2: "happy"}

return mood\_map.get(mood\_index, "unknown")

## FastAPI

FastAPI är ett modern Python-baserat ramverk för att bygga API-tjänster. Det är känt för att vara snabbt och enkelt att använda, och det stödjer asynkrona operationer, vilket är viktigt för att skapa responsiva och snabba tjänster. I projektet används FastAPI för att hantera all kommunikation mellan applikationen och DL-modellen. När en ljudfil skickas från desktop-applikationen, tar FastAPI emot ljudet, bearbetar det och skickar det till modellen för analys. Därefter returnerar FastAPI modellens förutsägelse tillbaka till applikationen, där den används för att generera en Spotify-rekommendation.

**FastAPI-initialisering**: Du initierar en FastAPI-instans med app = FastAPI(), vilket sätter upp en applikation som kan hantera REST API-anrop.

**Definierade Endpoints**:

* **Root Endpoint**: @app.get("/") är en GET-endpoint som returnerar ett meddelande för att verifiera att API

fungerar.

* **Analyze Voice Endpoint**: @app.post("/analyze-voice/") är en POST-endpoint som tar emot en ljudfil för analys. Denna endpoint:
  + Tar emot en fil som laddas upp via UploadFile.
  + Processar ljudfilen med hjälp av DL-modellen.
  + Returnerar ett JSON-svar som innehåller det förutspådda humöret och en Spotify-spellista baserat på detta humör.

Python

from fastapi import FastAPI, File, UploadFile

import librosa

import tensorflow as tf

import numpy as np

app = FastAPI()

# Ladda modellen

model = tf.keras.models.load\_model("combined\_pipe\_rnn\_modell\_saved")

# Endpoint för att analysera röst och förutsäga humör

@app.post("/analyze-voice/")

async def analyze\_voice(file: UploadFile = File(...)):

audio\_data, sr = librosa.load(file.file, sr=22050)

mood = predict\_mood(audio\_data)

playlist = get\_playlist\_for\_mood(mood)

return {"mood": mood, "playlist": playlist, "message": "Voice analyzed successfully!"}

**FastAPI som REST API-tjänst**: Eftersom FastAPI används för att hantera HTTP-begäranden och svar (i form av JSON-data), så uppfyller det kraven för ett REST API. Varje endpoint är en REST-resurs som kan anropas över HTTP med POST eller GET och returnerar svar i JSON-forma

## APPLIKTION

Tkinter används för att bygga desktop-applikationen som spelar in och skickar ljuddata till FastAPI-servern, samt visar den förutspådda känslan. Användaren kan spela in sitt röstljud via applikationen, som sedan bearbetar och analyserar ljudet och visualiserar resultatet i form av en känsloprediktion och en rekommenderad Spotify-spellista.

Python

import tkinter as tk

import requests

API\_URL = "http://127.0.0.1:8000/analyze-voice/"

def analyze\_voice():

with open("recording1.wav", "rb") as f:

files = {"file": f}

response = requests.post(API\_URL, files=files)

result = response.json()

mood = result.get("mood")

playlist = result.get("playlist")

output\_window.insert(tk.END, f"Detected Mood: {mood}\nPlaylist: {playlist}\n")

# Tkinter GUI inställningar

window = tk.Tk()

output\_window = tk.Text(window)

record\_btn = tk.Button(window, text='Record and Analyze', command=analyze\_voice)

record\_btn.pack()

output\_window.pack()

window.mainloop()

## Spotify-integration

Spotify erbjuder ett API som gör det möjligt för utvecklare att interagera med dess plattform för att skapa och rekommendera spellistor baserat på specifika ingångar. I projektet används Spotifys API för att hämta en spellista som matchar det förutsagda humöret. För närvarande genererar systemet en länk till en passande spellista, men fullständig automatisering av spellistuppspelningen planeras som en framtida utveckling.

En bild som visar text, skärmbild, Teckensnitt, design

Automatiskt genererad beskrivning

# Metod

## Systemarkitektur

Systemet består av tre huvudsakliga komponenter: ystemet består av tre huvudsakliga komponenter:

1. **Tkinter-applikation**: Användarens gränssnitt för att spela in ljud och se resultat.
2. **FastAPI-server**: REST API som hanterar kommunikation mellan klient och modell.
3. **DL-modell**: En CNN-modell som klassificerar ljuddata i tre känslotillstånd.

+-----------------------+ +----------------+ +----------------------+

| Tkinter-Applikation | -----> | FastAPI | -----> | DL-modell (TensorFlow) |

| (Skickar ljudfil) | | (Hantera API) | | (Förutsäger humör) |

+-----------------------+ +----------------+ +----------------------+

## Test och utvärdering

Systemet testades med olika röstinspelningar. Modellen visade en noggrannhet på 95,13 % med en förlust på 0,2650, vilket indikerar en hög tillförlitlighet.

# Resultat

## Systemfunktionalitet

Systemet lyckas integrera en DL-modell med desktop-applikationen och REST API

för att förutse humör från röstinspelningar och rekommendera en relevant Spotify-spellista. Vid en inspelning skickas ljuddata från applikationen till FastAPI-servern. Där bearbetas ljudet och analyseras med hjälp av DL-modellen för att identifiera humöret. Detta humör matchas sedan med en Spotify-spellista, och länken till spellistan skickas tillbaka till användaren.

För att illustrera funktionaliteten i FastAPI och Tkinter-applikationen, visas här ett exempel på ett JSON-svar från API:t:

Python

{

"mood": "calm",

"playlist": "spotify:playlist:YOUR\_CALM\_PLAYLIST\_URI",

"message": "Voice analyzed successfully!"

}

Detta svar möjliggör att applikationen kan visa en direkt visuell återkoppling till användaren om deras humör och ge en länk till en rekommenderad Spotify-spellista.

## Modellprestanda

DL -modellen uppnådde en noggrannhet på 95,13 % med en förlust på 0,2650, vilket indikerar en hög tillförlitlighet i känsloigenkänningen. Vid testning med olika röstinspelningar visade modellen god förmåga att särskilja mellan de tre känslotillstånden: lugn, energisk och glad. Prestandan är tillräcklig för praktiska tillämpningar där mindre avvikelser i klassificeringen kan accepteras. Resultaten visar att MFCC-funktionerna i kombination med CNN-modellen ger en robust metod för känsloigenkänning från ljuddata.

## Användargränssnitt

Tkinter-applikationen erbjuder ett enkelt och användarvänligt gränssnitt där användaren kan spela in sitt röstljud och få en omedelbar analys av humöret, samt en länk till Spotify-spellistan. Applikationen visualiserar resultatet genom att ändra bakgrundsfärg baserat på det identifierade humöret (t.ex. gul för glad, grå för neutral) och ger en direkt återkoppling till användaren. Designen av gränssnittet är enkel men effektiv, och användartester visade att det upplevdes som intuitivt.

# Diskussion

## Diskussion

I detta avsnitt diskuteras de centrala frågeställningarna i projektet och svaren som framkommit genom utvecklingen av systemet för känsloigenkänning och Spotify-integration.

**Hur kan DL effektivt integreras i en desktop-applikation för realtidsanalys av ljud?**

För att integrera Deep Learning (DL) i en desktop-applikation på ett effektivt sätt krävs ett robust kommunikationssystem mellan DL-modellen och användargränssnittet. I detta projekt har FastAPI använts som en mellanliggande server för att hantera förfrågningar och svar mellan applikationen och DL-modellen. Genom att använda FastAPI som ett REST API kan applikationen enkelt skicka ljuddata i realtid till modellen och ta emot svar snabbt. Denna metod gör det möjligt för DL-modellen att köras som en API-tjänst, vilket i sin tur möjliggör effektiv dataöverföring i realtid. För att minska latens används asynkrona operationer i FastAPI, vilket förbättrar responstiden. Tkinter har använts för att skapa ett användarvänligt gränssnitt där användaren kan skicka inspelat ljud till modellen och få ett direkt svar i form av en humörprediktion och en Spotify-spellista som matchar det förutspådda humöret.

**Vilka tekniska krav behövs för att säkerställa att ljudinspelningar kan analyseras och klassificeras med hög precision i realtid, och hur kan DL-modellen optimeras för att fungera effektivt i en desktop-miljö?**

För att uppnå hög precision och snabb respons i realtid kräver systemet både optimerad modellarkitektur och lämplig hårdvara. Modellen som används i detta projekt är ett CNN-nätverk (Convolutional Neural Network) tränat på ett omfattande dataset för att identifiera olika känslotillstånd från ljuddata. Modellen uppnådde en noggrannhet på 95,13 %, vilket indikerar att modellen är vältränad för detta specifika ändamål.

För att optimera modellen för användning i en desktop-miljö används TensorFlow, vilket tillåter CPU-baserad inferens utan att kräva dedikerade GPU- eller molnlösningar. Genom att begränsa storleken på MFCC-funktionerna som extraheras med hjälp av Librosa och använda en relativt lätt CNN-arkitektur kan modellen köras med låga resurskrav, vilket gör systemet lämpligt även för datorer med standardhårdvara.

**Vilka utmaningar uppstår vid länkning av humörigenkänning med en extern tjänst som Spotify?**

Vid integrering av humörigenkänning med en extern tjänst som Spotify mötte projektet flera utmaningar, särskilt inom autentisering och API-begränsningar. Spotify kräver OAuth 2.0-autentisering, vilket innebär att systemet behöver hantera tokens för att få tillgång till Spotify-innehåll. Denna hantering av tokenuppdateringar och autentisering gör det komplext att fullt automatisera Spotify-uppspelningen. Dessutom har Spotify specifika begränsningar på antalet API-anrop som kan göras per minut, vilket kan påverka användarupplevelsen om flera förfrågningar görs samtidigt.

För närvarande genererar systemet en länk till en relevant Spotify-spellista baserat på det förutspådda humöret, vilket kräver att användaren manuellt öppnar Spotify-länken. Framtida förbättringar inkluderar att automatisera själva uppspelningen, vilket kräver ytterligare arbete med Spotify API och dess autentiseringsprocesser.

**Hur kan kommunikation mellan FastAPI och Spotify effektiviseras för att säkerställa snabb rekommendation och uppspelning av spellistor som speglar användarens känslor?**

För att förbättra kommunikationen mellan FastAPI och Spotify, och därmed minska svarstider, kan systemet använda cache-lösningar. Genom att bygga ett cachelager i FastAPI-servern kan de mest frekvent använda spellistorna lagras temporärt. Detta minskar antalet anrop till Spotify och optimerar prestandan genom att spara tidigare genererade resultat.

En ytterligare optimering är att bara göra API-anrop till Spotify vid förändrade känslotillstånd, snarare än vid varje användarförfrågan. Genom att implementera en logik som registrerar tidigare humörprediktioner kan systemet undvika överflödiga API-anrop och därmed optimera datatrafiken.

**Hur kan systemet utökas för att förbättra användarupplevelsen?**

Systemet kan utökas och förbättras på flera sätt för att ge en mer omfattande och användaranpassad upplevelse. Genom att inkludera fler känslokategorier, såsom sorg, ilska och förvåning, skulle systemet kunna ge en mer nyanserad analys av användarens humör. Detta kräver dock ytterligare träning av modellen på ett bredare dataset som innehåller fler känslomässiga variationer.

För att ytterligare förbättra användarupplevelsen skulle systemet kunna göras molnbaserat, vilket skulle möjliggöra tillgång från flera enheter och även möjliggöra skalbar användning för ett större antal användare. Ett molnbaserat FastAPI skulle också möjliggöra integration med andra applikationer och enheter, inklusive röstassistenter, vilket skulle ge en hands-free-upplevelse där användaren kan aktivera och använda systemet med röstkommando

## Framtida utveckling

Vidareutveckling kan omfatta följande förbättringar:

* **Automatisering av Spotify-uppspelning**: För närvarande genereras endast en länk till spellistan. En framtida utveckling skulle kunna inkludera en direktuppspelning av Spotify-spellistan utan att användaren behöver öppna webbläsaren manuellt.
* **Utökat känsloigenkänning**: Genom att inkludera fler kategorier av känslor, till exempel sorg eller förvåning, kan systemet erbjuda en mer nyanserad analys. Detta skulle kräva ytterligare modellträning och en större mängd känslomärkta ljuddata.
* **Förbättrad modell och optimering**: För att göra modellen snabbare och mindre resurskrävande kan alternativa DL-arkitekturer, som kombinerar CNN och RNN, utforskas. Detta kan leda till en förbättrad användarupplevelse med kortare bearbetningstid.
* **Molnbaserad server**: Att driva FastAPI-servern och modellen på en molnplattform skulle möjliggöra åtkomst för en bredare användargrupp, vilket skulle öppna för att fler kan använda applikationen samtidigt och från olika enhete

# 6. Slutsats

* Detta projekt har visat hur DL kan integreras i en desktop-applikation för att analysera röst och identifiera användarens humör. Med hjälp av en REST API-tjänst skapad med FastAPI kommunicerar applikationen med DL-modellen, vilket möjliggör realtidsanalys och rekommendering av Spotify-spellistor. Systemet uppnår en hög grad av noggrannhet i känsloigenkänning, och den automatiska genereringen av spellistor som matchar humöret ger användaren en personlig musikupplevelse.
* Systemets nuvarande funktionalitet visar potentialen för vidareutveckling inom musikrekommendationer baserade på känslor, särskilt genom DL-modellens förmåga att upptäcka känslomönster från ljud. En av de största utmaningarna som kvarstår är att förbättra Spotify-integrationen så att uppspelningen sker automatiskt.

# 6. Självutvärdering

Jag tycker jag kan få G.

I mitt grupparbete, fungerade okej med kommunikation, men jag tycker ändå vi behöver ställa frågor bättre om inte förstår vad andra pratade om. I diskussion var bra, alla är aktiv på möten, på något sätt kommer vi var överens det bästa sett för projekt.

Jag hoppade in i gruppen en vecka efter de hade börjat. Så jag fick lära mig både hur skapa man desktop-applikation och vara med tränat dataset. Men det som jag lärt mig ordentligt är den sista steg när vi skulle interagera. Jag trodde från början att det är bara koppla och skriva ihop alla tre delar. Sen i mötet då förstod jag att det inte var det som jag trodde, och ingen hade aning att hur ska vi göra. Då tittade jag massor videor och dokument hur ska interagera. Sen parenterade till andra i gruppen, till sist lyckades vi med integrationen.

# 7. Källor

Tom Ash, “Machine learning is getting BIG (Part I),” Speechmatics, 2020. https://www.speechmatics.com/resources/articles-and-news/machine-learning-isgetting-big-part-i

R.Moutafis, “Why 90 percent of all machine learning models never make it into production,” Towards Data Science, Nov. 08, 2020. https://towardsdatascience.com/why-90-percent-of-all-machine-learning-models-never-make-it-intoproduction-ce7e250d5a4a (accessed May 09, 2022)

N. Ngo, “Deploying a Machine Learning Model as a REST API,” Towards Data Science, Aug. 2018, Accessed: Mar. 12, 2022. [Online]. Available: https://towardsdatascience.com/deploying-a-machine-learning-model-as-a-rest-api4a03b865c166

J. Brownlee, “How to Save and Load Your Keras Deep Learning Model,” Machine Learning Mastery, May 2019, Accessed: May 22, 2022. [Online]. Available: <https://machinelearningmastery.com/save-load-keras-deep-learning-models/>

<https://www.google.com/search?q=%E6%B7%B1%E5%BA%A6%E5%AD%A6%E4%B9%A0%E4%B8%8Efastapi+%E7%9A%84%E7%BB%93%E5%90%88&sca_esv=9665476702a22ff3&rlz=1CDGOYI_enSE1007HU1007&hl=sv&biw=375&bih=634&sxsrf=ADLYWILhYm6gLaPmOF1r8uFA6qCsVf5-mQ:1729181935162&ei=7zgRZ5bFCZixwPAP3MfM-AY&oq=%E6%B7%B1%E5%BA%A6%E5%AD%A6%E4%B9%A0%E4%B8%8Efastapi+%E7%9A%84%E7%BB%93%E5%90%88&gs_lp=EhNtb2JpbGUtZ3dzLXdpei1zZXJwIiDmt7HluqblrabkuaDkuI5mYXN0YXBpIOeahOe7k-WQiDIEECMYJ0iZWlCGDVi1DnAAeACQAQCYAW2gAbwCqgEDNC4xuAEDyAEA-AEBmAIDoALsAZgDAIgGAZIHAzIuMaAH7gg&sclient=mobile-gws-wiz-serp&tbm=vid#fpstate=ive&vld=cid:20ff4087,vid:NrarIs9n24I,st:0>

https://www.youtube.com/watch?v=vEwN7Mqn1MM&list=PLmvLOUyx8RK8E1OzDsDistfbT6-qumiRz

# Källförteckning