## Feladatkiírás

A hallgató feladata önfelügyelt módon előtanított (wav2vec 2.0) modellt tanítani, amely képes az érzelem felismerésére az alany beszédhangjából. Erre a szabadon elérhető MELD adatbázist kell használnia; a feladat része az adatbázis konvertálása (pl. mintavételezési ráta, annotáció formátuma) is. Mivel az érzelemfelismerési feladatokban jellemző, hogy az egyes érzelemosztályok gyakorisága között nagy különbség van, szükség esetén ezt is kezelnie kell.

## Tartalmi összefoglaló

* A téma megnevezése:

A megnevezés „Hang alapú érzelemfelismerő rendszer gépi tanulással a wav2vec2 technológia használatával”

* A megadott feladat megfogalmazása:

A wav2vec2-n alapuló modell alapján tanítani a MELD adatbázison egy gépi tanuló modellt, ami képes hang alapján felismerni a különböző érzelmeket.

* A megoldási mód:

Pytorchban Speechbrain modellen alapuló IEMOCAP-wav2vec2 modell alapján létrehoztam a MELD adatbázisra alapuló modellt, és azt tanítottam be.

* Alkalmazott eszközök, módszerek:

Pytorch, Speechbrain, wav2vec2, IEMOCAP

* Elért eredmények:

Az érzelmen tanult modell a teljes adathalmazon az error rate 52,4%, 10.8 a Macro F1, legkisebbre lecsökkentett adathalmazon 82,0% error rate, 16,7 Macro F1, második legnagyobb érzelemre lecsökkentetten pedig 73,9% error rate, 19,7 Macro F1. Az érzeten tanult modell pedig a teljes és a második legnagyobbra rendre a következőket érte el: 52,2% error rate, 32,5 Macro F1; 58,7% error rate, 30,3 Macro F1.

* Kulcsszavak:

Érzelemfelismerés, Speechbrain, MELD, Gépi tanulás, wav2vec2

## Tartalomjegyzék

[Feladatkiírás 2](#_Toc198990550)

[Tartalmi összefoglaló 3](#_Toc198990551)

[Tartalomjegyzék 4](#_Toc198990552)

[BEVEZETÉS 5](#_Toc198990553)

[1. GÉPI TANULÁS 6](#_Toc198990554)

[1.1 A gépi tanulás 6](#_Toc198990555)

[1.1.1 Felügyelt tanulás 6](#_Toc198990556)

[1.1.2 Felügyelet nélküli tanulás 7](#_Toc198990557)

[1.1.3 Félig felügyelt tanulás 7](#_Toc198990558)

[1.1.4 Önfelügyelt tanulás 7](#_Toc198990559)

[1.2 Neuronhálók 8](#_Toc198990560)

[1.3 Gépi tanuláson alapuló érzelemfelismerés 9](#_Toc198990561)

[1.3.1 *Hangfájl előkészítése a tanuláshoz* 9](#_Toc198990562)

[1.3.2 Az érzelem felismerés 10](#_Toc198990563)

[1.4 A Wav2vec2 technológia 10](#_Toc198990564)

[1.5 Speechbrain 11](#_Toc198990565)

[2. A MELD adatbázis 14](#_Toc198990566)

[2.1 Micsoda a MELD adatbázis? 14](#_Toc198990567)

[2.2 Statisztikák az adatbázisról 15](#_Toc198990568)

[2.3 A MELD felhasználása 18](#_Toc198990569)

[3. A MELD ADATBÁZISON BETANULT MODELLEM 19](#_Toc198990570)

[3.1 Kezdő lépések 19](#_Toc198990571)

[3.2 Eredmények 21](#_Toc198990572)

[3.2.1 Érzelmen való tanítás 21](#_Toc198990573)

[3.2.2 Érzeten való tanítás 31](#_Toc198990574)

[3.3 Konklúzió 34](#_Toc198990575)

[Irodalomjegyzék 35](#_Toc198990576)

[Nyilatkozat 37](#_Toc198990577)

[Köszönetnyilvánítás 38](#_Toc198990578)

# BEVEZETÉS

Az érzelem nekünk emberek számára kulcsfontosságú szerepet játszik a beszélgetések során. Képzeljük el, milyen lenne egy beszélgetés, ha minden monoton történne? Az érzelmek által sok mindent fejezünk ki magunkról és a szituációról, ami körülöttünk történik. Egy beszélgetés során sok mindennel meg tudjuk mutatni érzelmeinket: hanglejtés az egyik ilyen tényező, vagy az arckifejezés, testtartás, kézmozdulatok, mondat összetétele. Ezen kívül az érzelmek más területeken is fontos szerepet játszanak. Legyen szó kapcsolatokról, az érzelmek döntik el mit gondolunk a másikról, lehet ez akár szerelem, harag, empátia. Döntésekben is befolyásolnak minket az érzelmeink. Nem tudunk mindig racionálisan gondolkodni minden döntésről, van amikor az érzelmek is közbe szólnak. Ilyen például egy negatív szituáció, amiben voltunk már, és a korábbi tapasztalataink alapján félelmet érzünk miatta, vagy furdal a lelkiismeretünk tőle. De akár pozitív hatással is lehetnek ránk az érzelmek. A szenvedély, a vágy, a kíváncsiság, mind mozgató rugó tud lenni céljaink elérésben. A tudományos világban régebben nem vették figyelembe, hogy milyen hatással vannak az érzelmek a döntéseinkre. A szakdolgozatomban az érzelmet veszem figyelembe, ezen belül is a hang alapján kimutatott érzelmet. Egy modellt fogok betanítani Pytorch-ban Speechbrain eszköztár segítségével a MELD adatbázison. Célom, hogy a modell a tanítások után pontosan megmondja, milyen érzelem volt az adott mondat.

# GÉPI TANULÁS

## A gépi tanulás

„A gépi tanulás a [mesterséges intelligencia](https://hu.wikipedia.org/wiki/Mesters%C3%A9ges_intelligencia) egyik ága, olyan rendszerekkel foglalkoznak, melyek tanulni képesek, azaz tapasztalatokból tudást generálnak. A gyakorlatban ez azt jelenti, hogy a rendszer példa adatok, minták alapján képes önállóan, vagy emberi segítséggel szabályszerűségeket/szabályokat felismerni/meghatározni. A rendszer tehát nem csupán betanulja „kívülről” a mintákat, hanem képes ezek alapján olyan általánosításra, ami alapján – a tanulási szakasz végeztével – ismeretlen adatokra vonatkozólag is „helyes” döntéseket tud hozni.” (*Forrás:* [*Wikipedia*](https://hu.wikipedia.org/wiki/G%C3%A9pi_tanul%C3%A1s))

A gépi tanulást sokféleképpen lehet tanítani külöböző feladattípusokra. Általában egy feladatra egy modellt tanítanak, de akár több feladatot is lehet tanítani egy modellre. [1]

A gépi tanulás fajtái: [1]

* Felügyelt tanulás
* Felügyelet nélküli tanulás
* Félig felügyelt tanulás
* Megerősítéses tanulás
* Önfelügyelt tanulás

### 1.1.1Felügyelt tanulás

A felügyelt tanulásban meg van adva, hogy milyen választ várunk el a géptől. Ennek leggyakoribb feladatja az **osztályozás**. Valamilyen adatokból álló fix méretű vektort adunk át neki, majd ezeket sorolja be előre adott osztályok valamelyikébe. Lehet ez például kép-, karakter-, szöveg-felismerés, predikciós analizmus, fordítás, képgenerálás, és még lehetne sorolni tovább. Nagyon tág a lehetőség tanulás terén, rengeteg mindenre be lehet tanítani, ha van hozzá megfelelő nagyságú adathalmazunk és megfelelően is oszlanak el az adathalmazban a különböző „szabályok”, különben túltanulás léphet fel. [2]

A másik feladat, amit meg szokott oldani az a **regresszió**. [3] A felügyelt tanuló modellek nagyon pontosak tudnak lenni, viszont rengeteg időt elvesz ez a tanulás, ha nagy adathalmazt szeretnénk használni, hiszen ezeket az adatokat megfelelően fel kell címkézni, ami által később be tud tanulni a modell. [4]

### 1.1.2 Felügyelet nélküli tanulás

Felügyelet nélküli tanulásban semmilyen címkét nem adnak át a gépnek, ehelyett a gép talál meg összefüggéseket az adatok között, és nincsen szüksége emberi segítségre. Három feladata van, egyik ezek közül a **klaszterezés**. Itt különböző tulajdonságok alapján sorolja az adatokat egy-egy klaszterbe (csoportba). Például képeken az alapján tudna csoportosítani, hogy milyen közös dolog *- legyen ez akár tárgy, ember, állat -* látható a képeken. Másik feladat az **asszociáció**, itt az adatok és változók közt keres összefüggések a modell. Ennek egyik példája a marketingben az, hogy kiderítsék, melyik termékeket veszik általában közösen, ilyet lehet látni „Amazon”-on is. Az utolsó feladat pedig **dimenzió csökkentés**. Itt csökkenti a változók számát az adatokban, miközben próbálja nem rontani az adatokban által kinyert információkat. Erre egy jó példa a képeken a zajszűrés. A felügyelet nélküli tanulásban mivel nincsen szükség címkézésre, így nem vesz annyi előmunkát és időt igénybe egy ilyen modell betanítása. Viszont ez a modell nem tippel, mint a felügyelt tanuló modellek, hanem helyette csak csoportosítani tud hasonló adatokat. [3]

### 1.1.3 Félig felügyelt tanulás

A félig felügyelt tanulás egy köztes tanulási módszer a felügyelt tanulás és felügyelet nélküli tanulás között. Itt az adatbázis amit használunk tartalmaz előre felcímkézett és nem címkézett adatokat is. Ezzel a módszerrel pontosabb teljesítményt tudunk elérni például osztályozásokban és nem vesz annyi előmunkát és időt igénybe a címkézés, elég csak egy részét felcímkézni a tanításhoz. Ehhez egyik példa a CT képek vizsgálata, ahol már kevés címkézett tanító adattal már sokkal pontosabb eredményeket lehetne elérni a különböző rendellenességek megtalálására. [3]

### 1.1.4 Önfelügyelt tanulás

Az önfelügyelt tanulásban a tanítja saját magát címkézetlen adatokon, automatikusan generál hozzá címkéket, és felügyelet nélkül tanuló feladatokból csinál felügyelt tanuló feladatokat. Itt a generatív versengő hálózat segíti a modellt abban, hogy felismerje a hangfelvételben elhangzott szavakat. A **generatív versengő háló** egy mesterséges intelligencia modell amit két neurális hálózat alkot. Az egyik a generátor, ennek a feladata, hogy nem valódi – hamis - inputokat gyártson. A másik pedig a diszkriminátor, ennek a feladata pedig kitalálni, hogy az input amit kapott az valós vagy hamis. A generátor sok-sok iteráción megy keresztül, hogy végül olyan hamis inputot adjon, ami a diszkriminátor számára - *és akár számunkra is* - valódi példának tűnjön. [5]

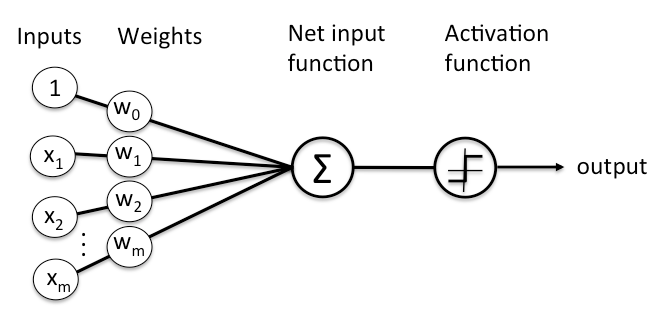
## 1.2 Neuronhálók

A neuronhálókkal próbálják szimulálni, hogyan működik az emberi agy. Különböző rétegek vannak, minden réteg sok neuronnal. [6]

A rétegek a következőképpen néznek ki: bemeneti réteg, rejtett réteg, kimeneti réteg. A bemeneti rétegen kapja meg az inputot, amit be kell majd sorolnia a megadott csoportok valamelyikébe. Az input réteg neuronjainak száma változó ez az input dimenziójától függ (például 28x28 pixeles kép esetében az input neuronok száma 784).

A rejtett rétegek száma eltérő, sok adat esetén a „mélyebb”, több rejtett rétegű háló tanul jobban. Ezzel hierarchikusan tudja feldolgozni az adatokat, és egyre komplexebb, egyre absztraktabb fogalmakat tanulnak meg, minél beljebb megyünk a rétegekben. Viszont a rejtett rétegek számával nő a számításigény is az algoritmusok miatt, amihez a videókártya minden erejére szükség van. [1]

A kimeneti réteg pedig az előre megadott csoportok, „label”-ek, amik közül egy osztályba kell besorolnia a modellnek a kapott input-ot. A rétegek egy irányba kommunikálnak egymással, súlyokat adnak át.



*1.2 1. ábra: Egy neuron számítása. Balról rendve haladva: bemenetek, súlyok, összesítő bemeneti függvény, aktivációs függvény, kimenet. Forrás: [7]*

Első lépésként az első rejtett réteg kapja meg az bemeneti réteg neuronjaitól az input-ot, majd ezt megszorozza mindegyik input súlyával, utána pedig összegzi ezeket a szorzatokat a saját „bias” értékével együtt. Ezek után ezt az eredményt tovább adjuk egy aktivációs függvénynek, ami kiszámítja az output-ját a neuronnak, ami egy értéket ad, ez aktivációs függvénytől eltérő, hogy milyen értéktartományban tud visszaadni értéket. Ezt megcsináljuk először az első rejtett réteg összes neuronjával, majd pedig ezt tovább adja hasonlóképpen a következő rejtett rétegnek, és így haladunk egészen a legvégén a kimeneti rétegig. [7]

## Gépi tanuláson alapuló érzelemfelismerés

A hang alapú gépi tanulásnak is számos fajtája van. Ezek közül néhány: osztályozás (ebbe beletartozhat a zenei hangszerek felismerése, zenei műfaj kitalálása, érzelem felismerése a beszélő személy felismerése), zajcsökkentés, upsampling. [8]

### 1.3.1 *Hangfájl előkészítése a tanuláshoz*

Nem olyan egyszerű a bemenet, mint egy képfelismerő modellnél. Ott a képet átadjuk, és ahány pixelje van a képnek, annyi neuron van az input rétegben. Viszont jön ekkor a kérdés: ezt hangoknál hogyan oldjuk meg?

A hangok analóg jelek, amik folyamatosan változó értékeket képviselnek, ehhez végtelen mennyiségű memória kellene tároláshoz. Az analóg jeleket konvertáljuk digitális jelekké, amik diszkrét értékeket vesznek fel. Ez a konvertálás kettő fő lépésből áll: mintavételezés (sampling), kvantálás (quantization). Mintavételezés során megadott időközönként veszünk egy-egy értéket az analóg jelből. A mintavételi frekvenciát Hertz-ben mérjük, ez határozza meg, hogy egy másodperc alatt mennyi értéket vegyünk fel. A nagyobb mintavételezési frekvencia során kevesebb mintavételezési hiba fordulhat elő, illetve hűebb eredményt ad vissza. A mintavételezési hiba, azért keletkezik, mert a folyamatos analóg jelet próbáljuk a véges számú mintaponttal leírni, ekkor viszont információ veszhet el. Ha van a hangban nagyobb Hz-ű komponens, mint amivel rögzítünk, akkor hibásan fogja visszaadni, torzulni fog a hang. A kvantálás során pedig a mintavételezés során kapott folyamatos értéket kerekítjük le egy véges számú diszkrét értékké. Ezeket az értékeket tudjuk digitális bitekként ábrázolni, ami a gép számára értelmezhető információ. Az értékeket kvantálási szintekhez rendeljük, és ezeket az értékeket kódoljuk bitekkel. Nagyobb bit mélységgel kevesebb kvantálási hiba fordul elő, illetve szélesebb a tartomány is, ezáltal a hangbeli különbségeket is jobban kezeli. CD-k esetében 16 bit-et használunk, ez 216 (65536) kvantálási szintnek felel A képen diagram, vázlat, minta látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.meg. [9]

*1.3.1 1. ábra: Mintavételezés és bit mélység közötti kapcsolat. Nagyobb bit számmal és mintavételi rátával pontosabb hangot kapunk vissza. Forrás: [10]*

### 1.3.2 Az érzelem felismerés

Miután digitális jellé alakítottuk át a hanganyagot, utána ezt tovább tudjuk adni egy neurális hálónak, és miután végig ment a rejtett rétegen, az utolsó kimeneti rétegben értékeket ad át a kimeneti neuronokba. A mi esetünkben nézve 7 érzelem közül fog tippelni, egy 1 dimenziós vektort visszadva, benne mind a 7 érzelem valószínűségével. Ha a valószínűségi vektor így néz ki: *[0.1, 0.2, 0, 0.2, 0.5, 0, 0],* az érzelmek meg így: *[NEU, ANG, JOY, SAD, DIS, FEA, SUR]*, akkor a 0.5 valószínűséggel tippeli meg a *DIS* érzelmet.

## A Wav2vec2 technológia

A wav2vec-et a META fejlesztette ki, 4,5 millió órányi hanganyagon tanult be. Szerintük azért fontos ez a beszédfelismerő AI, mert sok embernek nem az angol a fő nyelve, és ezáltal nem biztos, hogy tökéletesen tudja az angol szavakat kiejteni, vagy nem is tud angolul. Ehelyett ez a modell felcímkézetlen adatokból tanul, kevés vagy egyáltalán semmilyen leírt adatra nincsen szüksége. Ez az önfelügyelt tanulás a kulcsa annak, hogy ne csak egy nyelvet tudjon felismerni. Beszédegységekre tagolja a felvételt, amelyek nagyjából megfeleltethetők az egyes hangoknak. 7000 nyelvet beszélnek világszerte, például kirgiz nyelv is olyan, amiből nagyon kevés a hang alapján leírt szöveg. Címkézés alkalmazása nem egyenlő az emberi tanulással. A kiskorúak is úgy tanulnak nyelvet, hogy hallgatják körülöttük a felnőtteket beszélni. Ezáltal tanulják meg a kiejtést, nő a szókincsük. Ennek példája alapján próbálják a wav2vec2-t is tanítani. A modell 25 ezredmásodperc hosszúságú alapegységeket tanul meg, hogy képes legyen magasabb szintű, kontextusfüggő reprezentációkat elsajátítani. Ezzel olyan beszédfelismerő rendszert tudnak fejleszteni, ami a legjobb félig felügyelt módszereknél is jobban teljesít, akár százszor kevesebb tanítóadat felhasználásával. [11, 12]

Ennek az újabb verziója a wav2vec 2.0, amit én is használni fogok. Ez az új modell tanítva van arra, hogy megmondja helyesen a maszkolt (kitakart) hang egységet és közben meg is tanulja, hogy mik lehetnek azok az egységek. 10 perc feliratozott beszéddel és 53000 óra feliratozatlan beszéddel már képes a wav2vec2 beszédfelismerő modelleket tanítani 5,2%-os hibaaránnyal tiszta beszéden, illetve 8,6%-al zajos beszéden. Ezzel nem kell annyi feliratozott hanganyag, mint korábban. [13]

Kifejlesztettek egy új nyelvközi megközelítést is, ez az XLSR (Cross-Lingual Speech Representation). Az XLSR képes több nyelv közös beszédegységeit megtanulni. Ez akkor hasznos, amikor egy adott nyelven nincsen sok címkézetlen hanganyag, mivel azok a nyelvek is profitálhatnak, ahol több adat áll rendelkezésre. Mindössze 1 órányi felcímkézett adat tanításával a wav2vec 2.0 felülmúlja az előző csúcsteljesítményt a LibriSpeech benchmark 100 órás részhalmazán, úgy, hogy százszor kevesebb felcímkézett adatot használ fel. [13]

A modell úgy dolgozza fel az audió fájlt, hogy a nyers hanghullámot dolgozza fel egy többrétegű konvolúciós neurális hálóval, 25ms nagyságú szeletekre bontja a hangot. Ezeket a szeleteket tovább küldi egy kvantálóra és egy transzformer modellre. A kvantáló kiválaszt egy beszédegységet az előre megtanult egységek közül, amelyik legjobban illik az adott hangrészlethez. A hangszeletek felét kb. maszkolják, mielőtt tovább küldik a transzformernek. A transzformer összefüggéseket keres az egész audiószekvencián belül, hogy a maszkolt részeket jobban ki tudja következtetni. Utána pedig a modellnek meg kell mondania, hogy melyik kvantált egység tartozik a maszkolt helyekre. Ezzel megtanulja a modell belső szerkezetét anélkül, hogy leírásra lenne hozzá szüksége. [13]

## Speechbrain

Speechbrain egy open-source minden-az-egyben beszédfeldolgozó eszköztár. Felhasználóbarát eszköztárat akartak alkotni, amivel egy kis Python tudással már szinte bárki használhatja valamelyik beszédfelismerési technológiát. Sok képzési sablonja és előre betanított modellje van népszerű adatbázisokhoz. Emellett jól dokumentált és oktatóanyagok is készültek a használatukhoz. Mivel open-source, így nyugodtan alkothat bárki hozzá, illetve fejlesztheti az eddigi munkákat. Olyan tanító adatokat használnak, amiből a legtöbb publikusan elérhető, ezen kívül az eredményeket is megosztják, így bárki el tudja érni ugyanazokat az eredményeket mint ők. Pytorch-ot használva alkották meg ezt az eszköztárat, ami az egyik fő gépi tanuló könyvtár Pythonban. A Pytorch kódja emellett kompatibilis a Speechbrain-ben is, nem kell megváltoztatni emiatt a program kódját. Minden függőség, ami a Speechbrain-hez szükséges, az elérhető egyetlen ***„pip install speechbrain”*** paranccsal. Egyszerre több feladatot is képes megoldani a Speechbrain, például egyszerre tudja felismerni a beszédet, megérteni a beszéd kontextusát, nyelvét, érzelmét és beszélőit. Ezen kívül a beszéden tud javítani, ha például túl zajos lenne a hang, külön tudja választani a beszélők mondatait, ha egyszerre többen beszélnének. A fejlesztői remélik, hogy ezzel az eszköztárral jelentősen fel tudják gyorsítani a beszédfeldolgozás és mélytanulás területén végzett kutatásokat. A Speechbrain egy könyvár és egy framework között helyezkedik el, mert a Speechbrain moduljait könnyedén meg tudjuk hívni és az alapján “építkezni”, úgy, mint a könyvtárakban, viszont mint egy framework-nél, létre kell hoznunk egy osztályt, ami a Brain osztályból öröklődik. A Brain osztállyal tanítjuk be a modellünket, ebben már lényegében meg van írva a tanítás folyamata, ezzel nem kell foglalkoznunk, de kedvünkre alakíthatjuk is akár a tanítást. [14]

import torch , speechbrain

class SimpleBrain (speechbrain.Brain):

def compute\_forward(self, batch, stage):

return self.modules.model(batch["input"])

def compute\_objectives (self, predictions, batch, stage):

return torch.nn.functional.l1\_loss(predictions, batch["target"])

modules = {"model": torch.nn.Linear(in\_features=10, out\_features=10)}

brain = SimpleBrain(modules, lambda x: torch.optim.SGD (x, 0.1))

data = [{"input": rand(10, 10), "target": rand(10, 10)}]

brain.fit(epoch\_counter = range(15) , train\_set = data)

*1.5 1. kódrészlet: Egyszerű modell betanítása Speechbrain-ben Brain osztály használatával. Forrás: [14], Listing 3*

Ebben a kódrészletben a ***compute\_forward*** és ***compute\_objectives*** osztályban volt módosítás, az elsőben megadtuk, hogy amit kap adat - ami egy json itt -, abban az „input” kulcsot nézze, a másodikban pedig beállítjuk az L1 loss-t függvénynek, és itt pedig a „target” kulcsot nézze. Ez után beállítjuk azt is, hogy a modellnek a hálói hogyan épüljenek fel: 10 input neuron és 10 output neuron.

Ezen kívül egy .yaml fájlt is szoktak használni, ez egyenlő egy konfigurációs fájllal. Ebben a fájlban mondjuk meg, hogy mennyi epoch-on menjen végig a program, mekkora legyen a batch méret, hova mentse a modellt, hol találja az annotációkat, milyen osztályt használjon kimeneti neuronoknál, milyen loss függvényt használjon, stb. Ez csak pár példa, ezzel próbáltam szemléltetni, hogy miket állítunk be ebben a konfigurációs fájlban. A Speechbrain nem pontosan sima YAML-t használ, hanem ennek egy kibővített változatát, amit HyperPyYAML-nek hívnak. Ez a HyperPyYAML lehetőséget ad leírásokra is, amivel objektumokhoz hozzájuk tartozó argumentumot is megadhatunk. Ezzel így egyértelmű hivatkozás van a definíciók és az azokat használó objektumok között, így a kód átláthatóbb és könnyebben debugolható lesz. [14] Miután kész van a yaml és python fájl is, utána elindíthatjuk a programot a következő paranccsal:

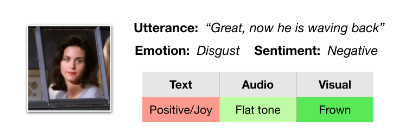
python train.py hparams.yaml

Itt értelemszerűen a train.py és hparams.yaml helyére azokat a neveket írjuk, ahogy elneveztük őket a programunkban.

# 2. A MELD adatbázis

## 2.1 Micsoda a MELD adatbázis?

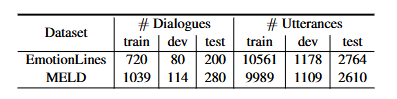
A Multimodal EmotionLines Dataset – más néven MELD – az EmotionLines kiterjesztése. 13000 mondat van benne 1433 dialógusból, mind a ***Friends*** TV sorozatból. Mindegyik mondat meg van jelölve a 7 érzelem közül egyel, illetve egy érzettel (sentiment). Az érzelmek: düh, undor, félelem, boldogság, semleges, szomorúság, lepődött. Az érzetek: pozitív, negatív, semleges. A MELD adatbázist azért hozták létre, mert az AI elterjedtével egyre nagyobb hangsúly van az érzelemfelismerésen is. Ennek az oka például a dialógus generálás, felhasználó viselkedésének megértése, többfajta adat (hang, szöveg, kép) megértése egyszerre. Ezzel például a felhasználó számára kedvező választ tud adni a rendszer. Csak néhány olyan érzelemfelismerő rendszer van, ami párbeszédben képes megérteni az érzelmeket, és ezek is csak kétszemélyes párbeszédre képesek, nem többszemélyesre. Az EmotionLines csak szöveg alapú megértést tud, ezt bővítették ki azzal, hogy hang és kép alapján ismerjen fel érzelmet, illetve érzetet is adtak neki. Sok minden nehézséget okoz az érzelem felismerésben, például a kontextusbeli változás, a beszélő hangulatváltozása. Előfordulhat az is, hogy a beszélő mozdulatairól, arcáról lehet megmondani, hogy milyen hangulata van, erre egy szimpla szöveg nem lenne elegendő arra, hogy megállapítsuk. Nagyon komplex az, hogy milyen érzelmet akar a beszélő kifejezni, ezért is nehéz egy ilyen érzelmet felismerni a párbeszéd szövege alapján. Egy párbeszédben szerepet játszik a testbeszéd, a hanglejtés, szóhasználat. [15]



*2.1 1. ábra: Szemléltető példa a három különböző csatorna együttes fontosságáról. A zöld szín mutatja azt, mekkora szerepe volt az adott csatornának az érzelemhez és érzethez. Forrás: [15], Figure 2*

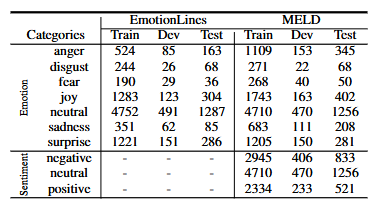
Példaként a felső képen a szöveg alapján azt mondanánk, hogy ez egy pozitív kijelentés. Viszont a hanglejtés és az arc alapján rájöhetünk, hogy ez szarkazmus. Még egy probléma a rövid, egyszavas mondatok. Egy „oké” vagy „nem” a kontextustól és a dialógus menetétől is függ. Körülbelül 42%-a az adatbázisnak kevesebb mint 5 szavas mondatokból áll. Az adatbázis a mondatok klipjeiből áll, illetve a .csv kiterjesztésű táblázatokból, amiben a le van írva szövegesen a mondat, meg van határozva a beszélő, az érzelem, érzet, és hogy melyik részben van benne és mikor.

## 2.2 Statisztikák az adatbázisról



*2.2 1 ábra: Táblázat az EmotionLines és MELD adatbázis dialógus- és mondat számáról Forrás: [15], Table 1*

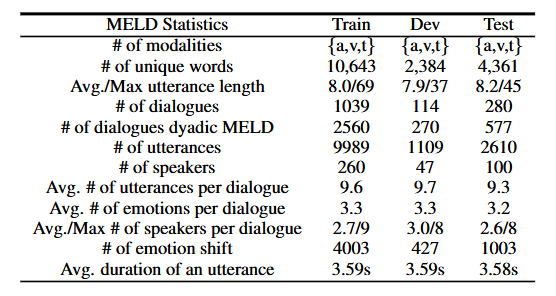
Ahogy már fentebb említettem, 13000 mondat van 1433 dialógusból. Ez szét van osztva 3 csoportra, ami a train, dev, test. Ahogy a képen is látszik, az EmotionLines-ban több mondat szerepelt, mert voltak például olyan dialógusok, amik több részből voltak összevágva egy dialógussá. Sok olyan dialógus is volt, ahol az érzelmet nem neutrálisnak nyílvánították, ami azt jelentette, hogy nem jutottak közös összhangra, hogy pontosan ez milyen érzelem lehet. Az ilyen érzelmeket is kivették az adathalmazból. Ami érdekesség, hogy ilyen dialógus, amiben nem jutottak összhangra az 2772 volt az EmotionLines-ban, de a vizuális és hang alapú dialógusok alapján már csak 89 ilyen volt, ebből is látszik, hogy a szöveg nem elég arra, hogy érzelmet felismerjünk vele. [15]



*2.2 2. ábra: Táblázat a különböző érzelmek és érzeteket mennyiségéről az EmotionLines és MELD adatbázisban Forrás: [15], Table 4*

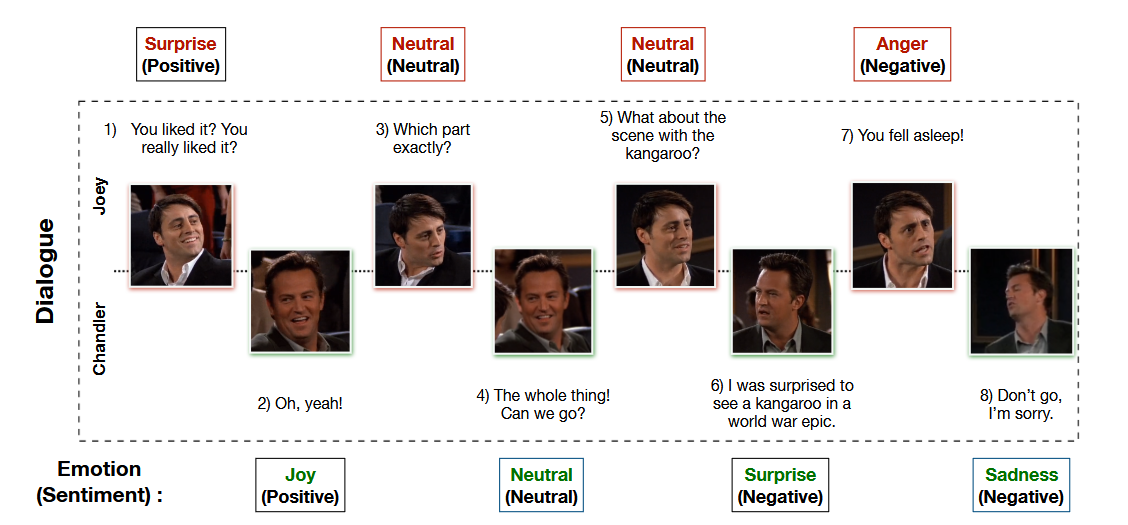
Néhány mondatnak az érzelme is megváltozott, a plusz kontextus alapján, amit a kép és hang adott egy-egy mondathoz. [15] Ahogy lehet látni a képen is, nagyon sok a semleges érzelem és érzet, mivel a való életben sem akar mindig valami érzelmet kifejezni a mondatával a beszélő. Legtöbb esetben csak egy állítás amiről beszélünk. Ezzel viszont az a probléma, hogy ami lehet, hogy a való életben hasonlóképpen történik, a tanításhoz nem éppen megfelelő. Ez miatt a programban overfitting léphet fel, amivel a sok különböző label-ű adat alapján általánosítani kezd a betanult modell. Viszont ezt nem olyan egyszerű megoldani, mert a modell méretének növelésével a rugalmassága is nő, így a tanítópéldákon egyre pontosabb lesz, a tesztpéldákon pedig egy ponton túl romlani fog a teljesítménye.

Az érzelmekre visszatérve, a lepődött érzelem egy példa a komplex érzelemre, mert egyszerre fejezhet ki pozitív illetve negatív érzetet is [15], ezért is jobb érzelem alapján label-ezni a példákat. Viszont az érzettel is hasonló a probléma mint az érzelemnél, mivel látszik, hogy a semlegesből nagyon sok van, a többi érzelemre pedig az arányok drasztikusak az érzelemhez nézve.



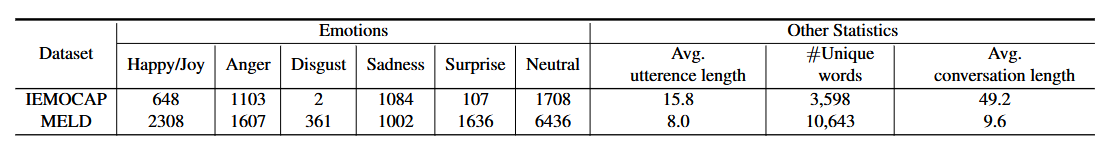
*2.2 3. ábra: Táblázat statisztikákkal a három csoport dialógusaira/mondataira Forrás: [15], Table 8*

Még egy érdekes statisztika arról, hogy mindegyik csoportban mennyi beszélő volt például, illetve hány szó szerepelt egy mondatban és hogy milyen hosszúak voltak. Ahogyan lehet látni a táblázaton, az átlag körülbelül ugyanolyan mindegyik csoportban. A mondatok hossza 8 szó körül mozog, míg a hosszuk körülbelül 3,59 másodperc átlagban. Ugyanez a közeliség igaz az érzelmekre is. Egy dialógus körülbelül 9 mondatból áll, ebből 3 mondatban érzelmet fejeznek ki. Ezáltal is az bizonyosul, hogy a semleges érzelemből van a legtöbb. Ahogy lehet látni a táblázatban is, gyakran előfordul az érzelemváltás is. Ez még jobban megnehezíti az érzelem felismerését.



*2.2 4. ábra: Kép arról, hogy két ember közötti beszélgetés során hogyan változik az érzelem és érzet. Forrás: [15], Figure 1*

A fenti képen egy beszélgetést lehet látni. Az egyik beszélő Joey, a másik meg Chandler, az arckifejezése és mondata Joey-nak felül látható, Chandler-é pedig alul. Ezek mellet látható a mondat érzelme, illetve zárójelek között az érzet is. 5-ször változott az érzelem a beszélgetés során, és mindig másik érzelemre. Ez egy jó példa arra, hogy tényleg gyakran változik az érzelem egy dialógus során és a lepődött lehet negatív illetve pozitív érzetű is.

*2.2 5. ábra: Táblázat az IEMOCAP és MELD közötti különbségekről. Forrás: [15], Table 10*

A fenti táblázaton az IEMOCAP-pel van összevetve a MELD. Az IEMOCAP egy másik híres adatbázis az érzelemfelismeréshez. Ahogy a táblázaton is látszik, a MELD-nek sokkal több az adatja hozzá képest, viszont a beszélgetések hosszabbak. Ez azért van, mert az IEMOCAP 10 órányi beszélgetésből áll, 10 beszélgető személlyel. [15] Itt az érzelmek a következők: düh, boldogság, szomorúság, semleges, izgatottság, frusztráltság. Ami még hátrány az IEMOCAP-nél, hogy csak egy az egyhez való beszélgetések vannak, ez is ronthat a model betanításán. [15]

## 2.3 A MELD felhasználása

Sok különböző felhasználása lehet a MELD-nek. Felhasználható például érzelmi osztályok betanítására, amelyeket később generatív párbeszédrendszerek érzelmi receptoraként lehet alkalmazni. Ezáltal a model fel tudná ismerni, hogy milyen érzelmet rejt, amit a felhasználó mondott és ez alapján tudna megfelelő választ generálni a robot - legyen ez például chatbot -, így empatikusabban tudnak reagálni, illetve jobban meg tudják mondani a felhasználó személyiségét is. A több adatfajta miatt a MELD-et lehet használni multimoduláris beszélgető rendszerek betanítására is. Ezeknek a rendszereknek hozzáférésük van a beszélő hangjához és képéhez, és ez segít nekik a válasz generálásában. Az ilyen modellek sokat segítenek a személyi asszisztensekben, mint például Siri az Apple telefonokban vagy Google Gemini az Android telefonokban. Azt viszont fontos megjegyezni, hogy ez az adatbázis magában nem elég nagy egy ilyen rendszer betanítására, de remélhetőleg, amilyen módszereket használtak a MELD adatbázis létrehozásában azt tudják alkalmazni másik példákon is, legyen szó például még egy sitcomról. [15]

# 3. A MELD ADATBÁZISON BETANULT MODELLEM

## 3.1 Kezdő lépések

A modell betanításához Speechbrain-t használtam, ami PyTorch alapú. Az elején gondokba ütköztem tanítás ügyileg, mivel a Pytorch-ban a tanítást AMD kártyán nehezebb megtenni. Amikor elkezdtem a programom írását, Windows-on nem volt támogatott az AMD PyTorch-al, hiába próbáltam WSL-t (Windows Subsystem for Linux) használni, rendes Linux alapú rendszer kellett használatához. Így Ubuntu-t választottam operációs rendszernek. Ubuntu-n a PyTorch használatához le kellett töltenem az ROCm-et (Radeon Open Compute), ami nyílt forráskódú, AMD által fejlesztett platform, nagy teljesítményű számításokhoz és GPU-gyorsított alkalmazásokhoz. Lényegében ez az NVIDIA CUDA-hoz hasonló, de AMD videókártyáira. Használják számításokhoz, szimulációkhoz, gépi tanuláshoz. Telepítése után Python-ban letöltöttem a PyTorch-hoz tartozó csomagokat, és teszteltem, hogy felismeri-e a videókártyámat. A videókártya, amit használtam tanulásra egy AMD RX 7800 XT volt, 16 GB memória van benne. Ezek után letöltöttem a Speechbrain könyvtárait is. Következő lépésként letöltöttem a MELD adatbázist. Ez tartalmazott három mappát (dev, train, test), mindegyikben az mp4 fájlok, illetve 3 csv, amiben a labelek voltak, illetve le volt írva a beszélő és a szöveg. Az .mp4 fájlokat kellett átkonvertálni .wav fájlokká mono 16kHz-re. Ehhez a *ffmpeg*-et használtam, a következő paranccsal:



*3.1 1. kódrészlet: Pythonban írt for ciklus, ffmpeg segítségével az összes .mp4-ből .wav fájlt konvertál*

Ez végigfutott az összes fájlon, és .wav kiterjesztésűvé konvertálta őket monoban és 16 kHz-re. Ezen túl kellett csinálni .json fájlt, amiben minden wav fájlnak meg volt adva az útvonala, hossza, érzelme, érzete. Ehhez egy python scriptet írtam ( ***Mellékletek: A.***), ami a fájlt beolvasva kinyeri a hosszát, az érzelmet meg az érzetet pedig a csv-ből olvasta ki.



*3.1 2. kódrészlet: Példaként egy adat a következőképpen néz ki: látható a fájl pontos neve, mint kulcs, ennek értéke pedig a következő kulcs-érték párok: a fájl elérési útvonala, a fájl hossza, ami az első fájl esetében 2.795 másodperc, érzelem, ami itt SAD, ami a szomorúság, utolsó pedig az érzet, ami NEG, ami negatívat jelent.*

Miután ezek megvoltak jöhetett a betanítás. A Speechbrain-nek volt .yaml (*Mellékletek: C.*) illetve egy python scriptje (*Mellékletek: B.*), ami előzőleg IEMOCAP betanítására készült, ezeket vettem alapul a saját módosításaimmal. A .yaml fájl az hasonló egy xml alapú fájlhoz, lényegében ez a konfigurációs fájlja a modell-nek. Ebben állítom be, hogy mi legyen a seed, mekkora batch méret legyen, milyen annotációs fájlokat olvasson be, milyen osztályokat használjon tanításra, optimalizálásra, kiértékelésre, stb. A python fájlban pedig a tanulás része van a modell-nek, hogyan tanuljon, mit csináljon minden batch elején végén, kiértékelés. Először nehézségekbe ütköztem, mert sokszor leállt a program kevés memóriára hivatkozva. Mint kiderült, voltak olyan fájlok, amik túlságosan hosszúak voltak, ezért rengeteg memóriát akartak igénybe venni. A leghosszabb fájl 41 másodperc volt, az első 5 másodperc, amiben elhangzik belőle bármilyen érdemi információ, és feleslegesen ilyen hosszú. Ezt töröltem a többi hasonlóval együtt, amik maradtak azok 20 másodperc alattiak voltak. Körülbelül 10 ilyen fájlt vettem ki az adathalmazból, szóval nem történt érdemi változás. A modell végül 7-es batch méreten tanult be a teljes adathalmazon, downsample-ölt halmaz a legkisebb érzelemre, illetve a második legnagyobb érzelemre.

## 3.2 Eredmények

### 3.2.1 Érzelmen való tanítás

Ahogyan fentebb már volt kép arról, hogy melyik érzelemből mennyi van, csináltam róluk diagramokat, ahol szemléletesebb, hogy mekkora a különbség van az érzelmek között.

*3.2.1 1. ábra: Train adathalmaz*

Az érzelmek címkéi a következők: NEU a semleges, SUR a lepődött, FEA a félelem, SAD a szomorú, JOY a boldog, DIS az undor, ANG a düh. Jól látszik, mekkora különbség van az érzelmek között arányaiban: majdnem annyi van a semleges érzelemből, mint a többiből összesen. Az arányok sem jók, ha azt nézzük, hogy a semlegesből 4707 van a train halmazban, a félelemből -ami a legkisebb- pedig 268.

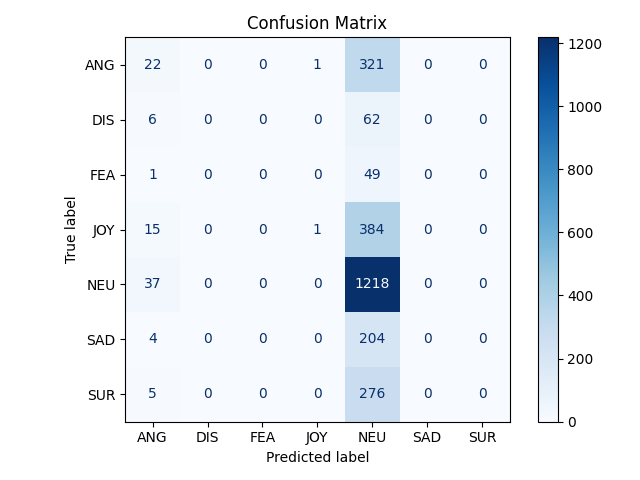
*3.2.1 2. ábra: Dev adathalmaz*

A dev halmazban is hasonlóak az arányok, semlegesből van a legtöbb, második legtöbb a boldogság, de ettől függetlenül is nagyon nagy a különbség köztük. Érdekes módon, itt az undorból sokkal kevesebb van, mint a félelemből, míg a train halmazban 3-mal volt több, mint a félelemből.

*3.2.1 3. ábra: Test adathalmaz*

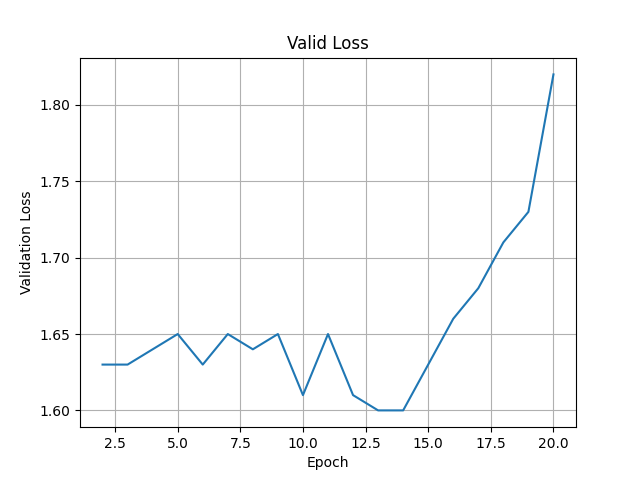
Itt is látszik, hogy a semleges érzelem a többihez képest rengeteg, hiszen a való életben sem érzelemteli minden kijelentésünk. Ezért is volt nehéz betanítani a modellt, nem nagyon akart tanulni, bármelyik módszerrel is próbálkoztam. Érdekes módon, ezen a nagy adathalmazon a validációs hiba ráta egyáltalán nem változott, az összes epoch során ugyanott maradt. Betanult a semleges adathalmazon, és minden mást is annak próbált betippelni. Dr. Gosztolya Gábor küldött nekem egy kiegészített osztályt, ami a következőket számolta az epochok végén: micro F1, macro F1, precision, recall (*Mellékletek: D.*). Ezen kívül csináltam egy függvényt, ami akkor fut le amikor a teszt halmazon megyünk végig (*Mellékletek: E.*). Ez függvény az adatokat kiírja egy .json fájlba, ami az összes tesztelt .wav fájlt leírja, hogy melyik labelt tippelte. Utána pedig egy másik függvénnyel ezeket a tippeket összekötöm azzal az annotációs fájllal, ahol meg van adva az eredeti label-je a mondatnak, ebből így kapok még egy .json-t, amiben pedig az igazi label van összevetve a tippeltel (*Mellékletek: A.*). Ebből számolok tévesztési mátrixot illetve normalizált mátrixot (*Mellékletek: F.*). Ezen kívül van még egy másik függvény, ami a train-log.txt-ben tárolt metrikák alapján diagramokat csinál az összes mért eredményről epoch szerint (*Mellékletek: F.*). Itt lejjebb van a tévesztési mátrix a teljes adathalmazon betanított modellről:

*3.2.1 4. ábra: Tévesztési mátrix a test adathalmazon*

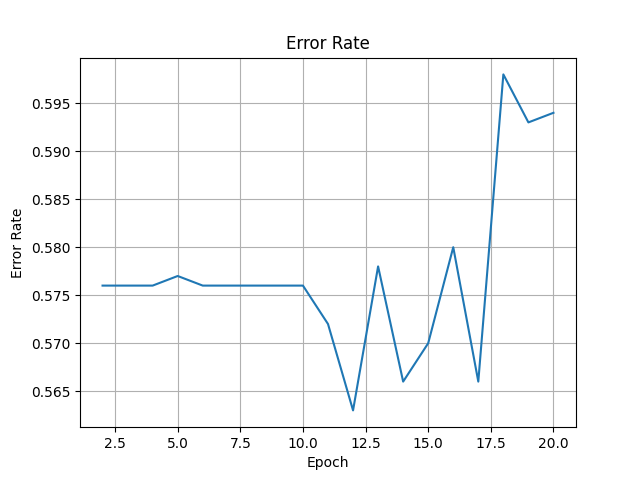


A mátrix a 12. epochon értékelődött ki, ezen volt a legkisebb az error rate. Ahogyan látszik a mátrixon, a legtöbbet semlegesnek tippelte, ezzel a legtöbbet eltalálta. Ezen kívül párat tippelt még dühnek. Furcsa, hogy csak a düht tudta megkülönböztetni a semleges érzelemtől, úgy is, hogy a boldogságból illetve a meglepődött érzelemből több adat volt, mint a dühből. Ebből esetleg arra lehetne következtetni, hogy a düh az, ami legjobban megkülönböztethető hangnem alapján. Ezeknél a hangnem sokkal feszesebb, hangosabb, ez miatt lehet jobban meg tudja különböztetni a többitől. Kettő boldogot is tippelt, abból egy jó is volt. Ezen kívül a többi érzelemre egyáltalán nem tippelt. A validáció és az error rate az epochok elteltével folyton romlott, de közben a precision, macro F1, recall értéke jobb lett:

*3.2.1 5. ábra: Veszteség a dev halmazon*



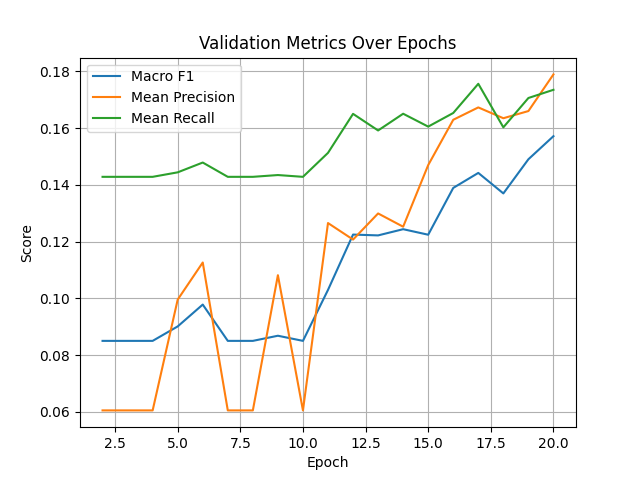
Ezen a képen is látszik, hogy az epoch-ok során egy ideig úgy tűnik, hogy tanul a modell, viszont a 14. epoch után ugrásszerűen elkezd romlani a modell. Ez azért történhet meg valószínűleg, mert egy idő után megtanulja a train halmaz válaszait, és nem akar tanulni már belőlük, vagyis overfitting lép fel. A veszteséget Negative Log-Likelihood függvénnyel számoljuk ki. Az NLL nem csak azt várja el, hogy jól tippeljen a modell, hanem hogy mennyire biztosan tippeli a modell a választ. A kisebb eséllyel eltalált válaszokat bünteti az NLL. [16]



*3.2.1 6. ábra: Error rate a dev halmazon*

Az error rate a következőképpen számítódik ki: a hibás predikciókat elosztjuk az összes predikcióval, és így kapjuk az arányt, hogy mennyit tippelt jól a modell. A legkisebb a 12. epoch-ban a 0.565, amit 56,5%-os hibázást jelent. Itt is látszik, hogy az utolsó epoch-okban nagyon megugrott az hibás tippek száma, ez is arra mutat, hogy overfitting lépett fel.

*3.2.1 7. ábra: Metrikák a dev halmazon*

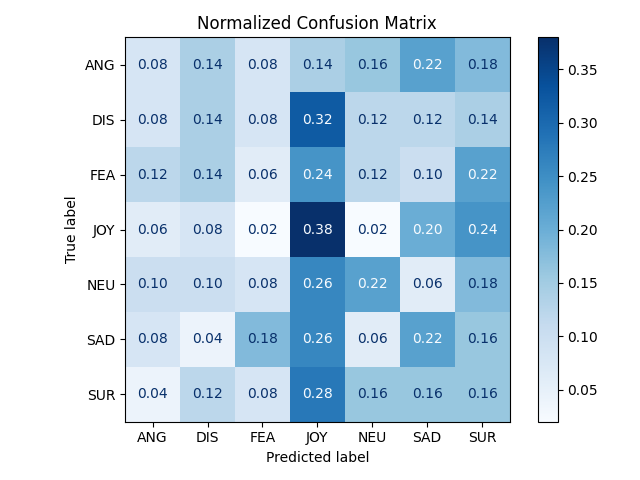


Ami egy ok lehet ezen metrikák növekedésére az az overfitting. Megtanulja a train adathalmaz adatait, és ezért ugrik ilyen nagyokat a valid loss és az error rate. Az F1, precision, recall növekedését eközben arra tudok tippelni, hogy egyre többet tippel semlegesnek, és mivel a semlegesből van a legtöbb, így ha abból többet tippel, több lesz a helyes érték. A többi kiértékelésben illetve itt is korábbi epoch-okon a legjobb az eredmény, ezeket külön lementi a legutolsó tanult epoch mellé. A legjobb amit lementett az a legkisebb error rate-ű epoch. Ez alapján csinál kiértékelést a teszt halmazon. Most nézzünk meg egy másikat, amiben a downsample-ve lett mind a három adathalmaz. A downsample úgy lett végrehajtva, hogy csináltam egy .json-t, amiben összegezve van, hogy mindegyik érzelemből mennyi van. Majd ezt felhasználva megnéztem melyik a legkisebb, és a többiből kidobtam annyit véletlenszerűen, hogy annyi maradjon bennük, mint a legkisebben (*Mellékletek: A.*). Itt vannak a diagramok erről:

*3.2.1 8. ábra: Downsample-ölt train halmaz*

Ahogyan lehet látni ezeken a diagramon is, jelentősen lecsökkent ezzel az adatok száma. 9983 volt a train halmazban, ebből lett 1876, ez 81%-os csökkenést jelentett. A többi halmazon is hasonlóképpen történt a változás, a dev halmazon is összesen lett 154 adat, itt mindegyik címkéből 22 maradt, míg a teszt halmazon 50 maradt minden címkéből, összesen 350 adat maradt. Ezzel a módszerrel jobb eredményeket ért el, mint a rendes, egész adathalmazon, de valószínűleg a kevés adat miatt nem is tanult be teljesen. Nagyon sokfelé tippelt, legtöbbet a boldogság érzelemnek hitte, így a boldogságból sokat el is talált. Ezen kívül a semlegest is jól tippelte. Ami érdekes adat még, hogy a meglepettet is sokszor keverte örömnek, és fordítva is. Ez lehet befolyása annak, hogy a meglepődött lehet negatív meg pozitív is, és amit pozitívnak gondolt, azt boldognak tippelte. Viszont ez az adathalmaz nyílván sokkal gyorsabban tanult, mint a rendesen. Egy epoch 1 perc volt, míg a másikon 10 percbe telt. A kevés tanulást az is bizonyítja, hogy a legjobb eredményt az utolsó, a 20. Epoch érte el. Itt vannak lejjebb tévesztési mátrixok a downsample-özött adathalmazról, érdekes eredmények születtek:

*3.2.1 9. ábra: Normalizált tévesztési mátrix a downsample-özött test adathalmazon*

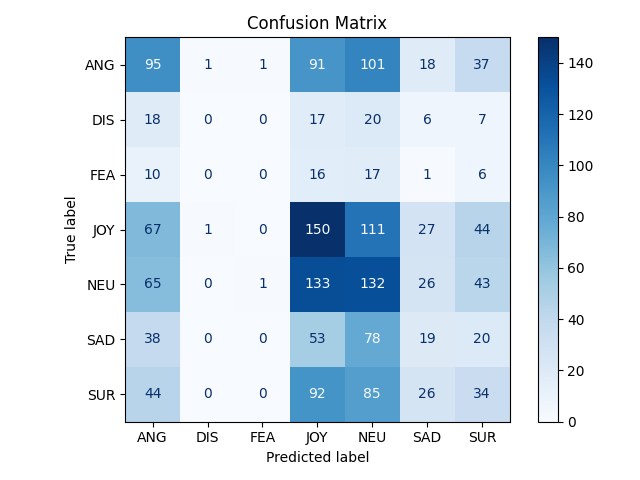


Ezután pedig próbálkoztam olyan változattal, ami csak a semleges érzelmet veszi le a második legnagyobbra (*Mellékletek: A.*). Ezzel sok adatot nem vesztünk, mint az előzőben, de így is nagy különbség van a semleges és a félelem között. Így a tanítás is csökkent újra, 6-7 perc körül volt egy epoch. Ezt a tanulást lehetne mondani a kettő másik tanítás ötvözetének is.

*3.2.1 10. ábra: A semleges adatok mennyisége lecsökkentve a második legnagyobb érzelemre (itt az öröm az)*

Ahogyan a grafikonon is látszik, a semleges-t a 4707-ről lecsökkentettük 1742-re. Minden más érzelemből ugyan annyi maradt. A semleges és a félelem között még így is nagy a különbség mennyiségre. Az örömre és a semlegesre állt be leginkább, de többet tippelt a többi érzelemre is, kivétel az undor és félelemre. A dühből is sokat eltalát jól, de sokat tippelt a dühből boldogságra és semlegesre is, illetve fordítva is: sok volt, ami semleges vagy öröm lett volna, de dühnek tippelte. Az undort és félelmet nagyon nem tippelte, mintha nem is létezne, a többi érzelembe próbálta tippelni. A szomorúságot és a meglepődöttet legalább kicsit megtanulta, volt abból olyan, amit jóra tippelt, de itt is sok volt az, ami a semlegesbe, boldogságba, dühbe került.

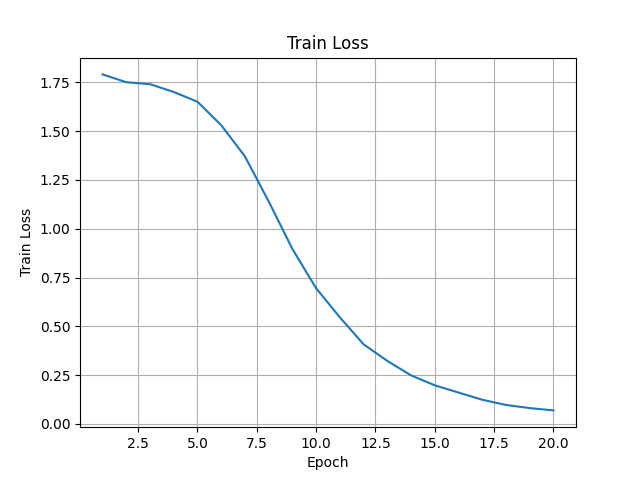
*3.2.1 11. ábra: Tévesztési mátrix a második legnagyobbra downsample-ölt teszt halmazon*



Ennél a modellnél egyébként a 6. Epochot értékelte ki, az lett a legjobb. Túlságosan betanulta a tanuló halmazt, ez miatt overfit lépett fel, megtanulta a training halmazt, de ezt ezután nem tudta alkalmazni az új adatokra. Nem tanulta a mintákat, helyette csak memoritzálta őket. Ezen is látszik, hogy az adat mennyiség így sem megfelelő, főleg úgy, hogy ekkora különbség van az érzelmek között.

A képen sor, diagram, Diagram, szöveg látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.



*3.2.1 12. ábra: Veszteség bal oldalt a train halmazon, jobb oldalt a dev halmazon. Semleges adatok száma lecsökkentve a második legnagyobbra.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | dev | | test | |
| Error Rate | Macro F1 | Error Rate | Macro F1 |
| teljes | 56,3% | 12,2507 | 52,4% | 10,7526 |
| ds | 75,3% | 22,9373 | 82,0% | 16,9992 |
| dsto2 | 69,0% | 24,1353 | 73,9% | 19,7304 |

*3.2.1 1. táblázat: Táblázat a dev és test halmazon elért eredményekről érzelemmel*

Egy táblázatba ki lett gyűjtve minden adathalmazra az error rate és a Macro F1. Ahol a legkisebb volt az error rate, ott futott le a test, illetve a dev adathalmazon is annak az epochnak az eredményeit gyűjtöttem ki. A Macro F1 az egy népszerű mérése a klasszifikációs modellek teljesítményére. [17] A Macro F1 ki kell először számolnunk a pontosságot (precision), illetve a visszahívást (recall). A precision-höz a képlet a következő:

*3.2.1 1. képlet: Precision képlete. Forrás: [17]*

A TP a true positive (igaz pozitív), az FP, ami false positive (hamis pozitív). Minden érzelemnek külön van precision-je meg recall-ja. Hogy pontosabb megértsük, micsoda pontosan a mi esetünkben a TP, FP, itt egy példa: a TP azok az eredmények, amit jól tippelt az adott érzelemre. Az FP az az, amit rosszul tippelt az adott osztályra. A recall képlete:

*3.2.1 2. képlet: Recall képlete. Forrás: [17]*

Itt annyi a különbség a pontossághoz képest, hogy a FP helyett itt FN-t használunk, ami false negative (hamis negatív), ami pedig az előző példából kiindulva: az adott érzelemhez tartozna a mondat, de a modell nem jól érzelmet tippelt. Miután a recall és precision megvan, minden osztályra kiszámolhatjuk az F1-et is. Ez pedig a következő:

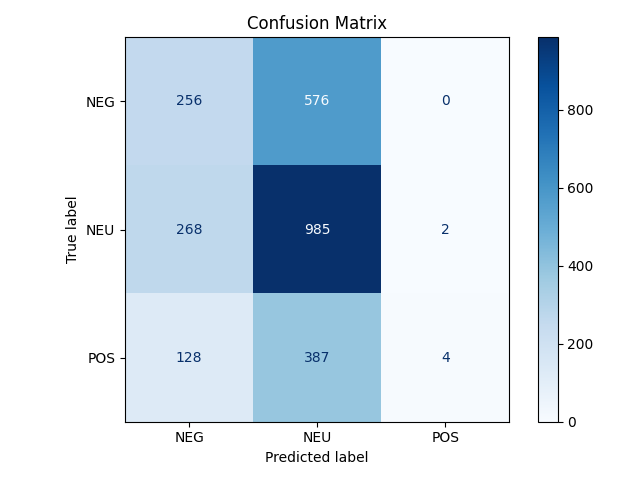
*3.2.1 3. képlet: F1 képlete. Forrás: [17]*

Miután megvan az F1 értéke is minden osztályra, utána számolhatunk belőle Macro F1 átlagot. Itt már csak az összes csoport F1 értékét összeadjuk és elosztjuk az osztályok számával. Ezzel az osztályok súlyozott átlagát nézzük. [17] Ahogyan a táblázatban is látni lehet (*3.2.1 1. táblázat*), a dsto2 volt a legeredményesebb tanulás szempontjából, itt a legjobb a Macro F1 a dev és test halmazon is. A legrosszabb eredményt a teljes adatbázison produkálta, valószínűleg a sok semleges adat miatt. Az error rate lehet, hogy jobb majdnem 13%-al a dsto2 adatbázison, viszont ez az overfitting miatt lépett fel, mert sok semlegest eltalált. Ezért is jobb a Macro F1 alapján nézni modellünk teljesítményét.

### 3.2.2 Érzeten való tanítás

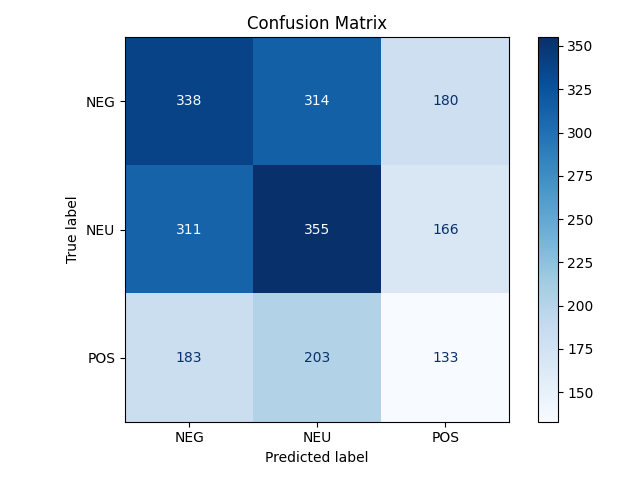
Ezek után megpróbáltam az érzettel is tanítani (*Mellékletek: G., H.*). Az adatbázis az ugyanaz, csak az érzelem label helyett az érzetet használtam. Itt három label van: semleges, pozitív, negatív. Az eloszlás jobb, mint a 7 darabnál, viszont a semlegesből így is jócskán több adat van a másik kettőhöz képest, kivéve a dev halmazon, ott nincs sokkal több a semlegesből, mint a negatívból. A train adathalmazon a semleges száma 4707, pozitív száma 2332, negatív száma 2944. A semleges és negatív között 1763 különbség van, ez azt jelenti, hogy 38%-al több a semleges adat, a pozitívhoz képest pedig 50%-al körülbelül.

*3.2.2 1. ábra: Tévesztési mátrix a teszt halmazon érzettel tanulva*



Először a teljes adathalmazon tanítottam be a modellemet. Ahogyan fentebb is írtam, a pozitívból volt a legkevesebb. Ez az eredményen is meglátszik. A 4. epoch volt a legjobb error rate kapcsán, a semlegesbe tippelte a legtöbb adatot, Sokat tippelt a negatívba is, a pozitívba lényegébe semmit, a adatból összesen 6-at tippelt oda. Itt is az lépett fel, hogy gyorsan megtanulta a train halmaz példáit, és nem tanult semmit belőlük az idő elteltével. Ebből is az látszik, hogy a majdnem 10000 adat kevés a tanításra. Hasonlóan állt be a rendszer tanítás után, mint az érzelmek tanítása után. A semlegesek száma nyílván ugyanannyi volt mint az érzelmeknél, a másik kettő eredmény ami más, de itt rendre mennyiség szerint változik az érzet alapján való tippelés. Ezek után megpróbáltam azt is, hogy a semleges méretét levettem a második legnagyobbra. Az eredményei ennek lejjebb láthatóak.

*3.2.2 2. ábra: Tévesztési mátrix a teszt halmazon érzettel tanulva, semleges mennyisége csökkentve a második legnagyobbra*



A train halmazban nyílván nagy volt az ugrás a semleges adatnál, viszont a másik kettőben nem volt hatalmas változás: a test halmazon 1255-ről 832-re csökkent a semleges adatok száma, míg a dev halmazon 470-ről 406-ra. A 3. epoch-ot értékelte ki a modell a test halmazon, itt is a legkisebb error rate alapján mentette le a legjobat. Itt jobb eredményt ért el, mint a rendes nagy adathalmaz tanítása során, hasonlóképpen, mint az érzelemnél, viszont így sem mondhatónak jónak, amilyen eredményeket produkált. Jobban össze-vissza tippelt, a semlegesből meg a negatívból sokat eltalált, de nagyon sokat tippelt emellett rosszul is. A pozitívak száma több lett, sok olyan pozitív is volt, amit rosszul tippelt. Ebből arra tudunk megint következtetni, hogy ez az adathalmaz nincs megfelelően elosztva tanításhoz, illetve nincs is elég adat hozzá.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | dev | | test | |
| Error Rate | Macro F1 | Error Rate | Macro F1 |
| teljes | 51,1% | 35,6146 | 52,2% | 32,5100 |
| dsto2 | 52,4% | 36,3758 | 58,7% | 30,3152 |

*3.2.2 1. táblázat: Táblázat a dev és test halmazon elért eredményekről érzettel*

Az érzetnél érdekesebb szituáció lépett fel. A dsto2 adatbázison a dev jobb eredményt ért el, mint a teljesen, de a test-en rosszabb volt a dsto2 Macro F1-e. Ettől függetlenül a dsto2-t lehet jobbnak mondani, mert itt valószínűleg jobban felismeri a különböző osztályokat.

## 3.3 Konklúzió

Ahogyan sokat említettem, az eredmények alapján látszik, hogy ez az adatbázis sajnos kevés a tanításra, illetve nincsenek is megfelelő arányban elosztva a label-ek. Értelemszerűen a való életben is egy beszélgetés során „semlegesen” beszélünk legtöbbször, viszont ez nem húzható rá egy tanító adathalmazra. Egy gépi tanuló modellnek csomó különböző adat kell, különben *overfitting* lép fel, és elkezd általánosítani a legtöbb adattal rendelkező label-hez. Volt, hogy olyan hibába ütköztem, hogy rosszul jegyezték fel az érzelmet, ezért azt se mondható biztosra, hogy teljesen helyes, amit tanult a modellem, ez miatt tévesen is tanulhatta meg az érzelmeket, nem ismerhette fel a mintákat. Viszont, amit alkotni akartak a MELD adatbázissal egy jó ötlet, hiszen az érzelmek felismerése nagyon fontos szerepet játszik az emberek közötti kommunikációra. Ezt átvinni mesterséges intelligenciával rendelkező rendszerekre nagy kihívással jár. Az emberi gondolkodásmód, lélek, érzelem játéka nincsen egy robotban, ezért ezt nehezebb is “lefordítani” egy gépnek. Viszont ebből is arra tudunk következtetni, hogy ami számunkra természetes, az nem olyan egyértelmű egy MI-nek, ami csak a 0-kat és 1-eseket “látja”. Az viszont, hogy felismerné például egy chatbot, hogy akivel beszél, az hogyan érez, és ezáltal empatikusabb válaszokat tudna adni, az egy nagyon jó eredménye lenne ennek a tanításnak.

## Irodalomjegyzék

[1] Chrystal R. China – IBM Technology: *Five machine learning types to know* <https://www.ibm.com/think/topics/machine-learning-types> (utoljára meglátogatva: 2025-05-22)

[2] Megyeri István – *Bevezetés a mélytanulásba* <https://drive.google.com/drive/folders/1YL7jlNHM0wrOCsT4xeUawIRX-FmpO_Gx> (utoljára meglátogatva: 2025-05-22)

[3] Martin Keen – IBM Technology: *Supervised vs. Unsupervised Learning* <https://www.youtube.com/watch?v=W01tIRP_Rqs> (utoljára meglátogatva: 2025-05-22)

[4] Dave Bergmann – IBM Technology: *What is self-supervised learning?* <https://www.ibm.com/think/topics/self-supervised-learning> (utoljára meglátogatva: 2025-05-22)

[5] Martin Keen – IBM Technology: *What are GANs (Generative Adversarial Networks)?* <https://www.youtube.com/watch?v=TpMIssRdhco&t> (utoljára meglátogatva: 2025-05-22)

[6] IBM Technology: *What is a neural network?* <https://www.ibm.com/think/topics/neural-networks> (utoljára meglátogatva: 2025-05-22)

[7] Pathmind: *A Beginner's Guide to Neural Networks and Deep Learning* <https://wiki.pathmind.com/neural-network> (utoljára meglátogatva: 2025-05-22)

[8] Valerio Velardo – The Sound of AI: *Audio Signal Processing for Machine Learning* <https://www.youtube.com/watch?v=iCwMQJnKk2c&t> (utoljára meglátogatva: 2025-05-22)

[9] Valerio Velardo – The Sound of AI: *Understanding Audio Signals for Machine Learning* <https://www.youtube.com/watch?v=daB9naGBVv4> (utoljára meglátogatva: 2025-05-22)

[10] Ahsen Jawed – Hollyland: *What is Sample Rate in Audio? Its Types and Impact on Sound*<https://www.hollyland.com/blog/tips/what-is-sample-rate-in-audio> (utoljára meglátogatva: 2025-05-22)

[11] Alexei Baevski, Steffen Scheider, Wei-Ning Hsu, Alexis Conneau, Henry Zhou, Ronan Collobert, Abdelrahman Mohamed, Michael Auli: *Wav2vec* <https://ai.meta.com/research/impact/wav2vec/> (utoljára meglátogatva: 2025-05-22)

[12] Baevski, A., Zhou, Y., Mohamed, A., & Auli, M. (2020). wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. *Neural Information Processing Systems*, *33*, 12449–12460. <https://proceedings.neurips.cc/paper/2020/file/92d1e1eb1cd6f9fba3227870bb6d7f07-Paper.pdf> (utoljára meglátogatva: 2025-05-22)

[13] Alexei Baevski, Alexis Conneau, Michael Auli: *Wav2vec 2.0: Learning the structure of speech from raw audio* <https://ai.meta.com/blog/wav2vec-20-learning-the-structure-of-speech-from-raw-audio/> (utoljára meglátogatva: 2025-05-22)

[14] Ravanelli, M., Parcollet, T., Plantinga, P., Rouhe, A., Cornell, S., Lugosch, L., Subakan, C., Dawalatabad, N., Heba, A., Zhong, J., Chou, J., Yeh, S., Fu, S., Liao, C., Rastorgueva, E., Grondin, F., Aris, W., Na, H., Gao, Y., . . . Bengio, Y. (2021, June 8). *SpeechBrain: a General-Purpose Speech Toolkit*. arXiv.org. <https://arxiv.org/abs/2106.04624> (utoljára meglátogatva: 2025-05-22)

[15] Poria, S., Hazarika, D., Majumder, N., Naik, G., Cambria, E., & Mihalcea, R. (2019). MELD: A Multimodal Multi-Party Dataset for Emotion Recognition in Conversations. *Information Systems Technology and Design, SUTD, Singapore*. <https://doi.org/10.18653/v1/p19-1050> (utoljára meglátogatva: 2025-05-22)

[16] Alfrick Opidi, Abhishek Jha – Neptune AI: *PyTorch Loss Functions: The Ultimate Guide* <https://neptune.ai/blog/pytorch-loss-functions#Negative-Log-Likelihood> (utoljára meglátogatva: 2025-05-22)

[17] Kenneth Leung – Kdnuggets: *Micro, Macro & Weighted Averages of F1 Score, Clearly Explained* <https://www.kdnuggets.com/2023/01/micro-macro-weighted-averages-f1-score-clearly-explained.html> (utoljára meglátogatva: 2025-05-22)

## Nyilatkozat

Alulírott Mészáros Zsombor programtervező informatikus Bsc szakos hallgató, kijelentem, hogy a dolgozatomat a Szegedi Tudományegyetem, Informatikai Intézet Számítógépes Algoritmusok és Mesterséges Intelligencia Tanszékén készítettem, programtervező informatikus Bsc diploma megszerzése érdekében.

Kijelentem, hogy a dolgozatot más szakon korábban nem védtem meg, saját munkám eredménye, és csak a hivatkozott forrásokat (szakirodalom, eszközök, stb.) használtam fel.

Tudomásul veszem, hogy szakdolgozatomat / diplomamunkámat a Szegedi Tudományegyetem Diplomamunka Repozitóriumában tárolja.

Dátum

2025. 05. 24. Mészáros Zsombor

Aláírás

## Köszönetnyilvánítás

Köszönet Dr. Gosztolya Gábornak, egyetemi konzulensemnek a segítségért a 2 félév során.