Contest - 2024

Gelegonya Gergő , Bakó András

2024.12

1 Röviden a folyamat :

Az első verziót lényegében teljesen vakon készítettük. Mi majdnem elejétől kezdve versenyben voltunk és ez az idő sok tapasztalattal látott el minket. Ebben az időszakban még felszínes tudásunk és képünk volt róla, hogy mit kéne megvalósítani. Az első verziót nagy részben gyakorlat alapján raktuk össze, kis segítséget kérve GPT-től. Ez azért volt baj, mert rossz irányba vitt minket, ellenben utólag hálás vagyok, mert míg másoknak működött a GPT által kiköpött kód, addig nekünk mindent végig kellett pörgetni mire rájöttünk mi is a probléma pontosan és hogy mi milyen hatással van az eredményre. Kétszer kezdtük teljesen a nulláról újra. Így összesen 3 verzió jött létre, amelyek mindegyikén sokat tanultunk.

1. verzió :

Sehogyse tudtuk lényegi működésre bírni, hiába lett 100 soros a kód. Itt még bőven nem értettünk semmit. Legelső körben vakon elkezdtünk játszani gyakorlatilag mindennel. Az alapelv az volt, hogy mivel nem tudjuk mit szeretnénk látni, ezért letesztelünk mindent. Az első konfigurációk így néztek ki:

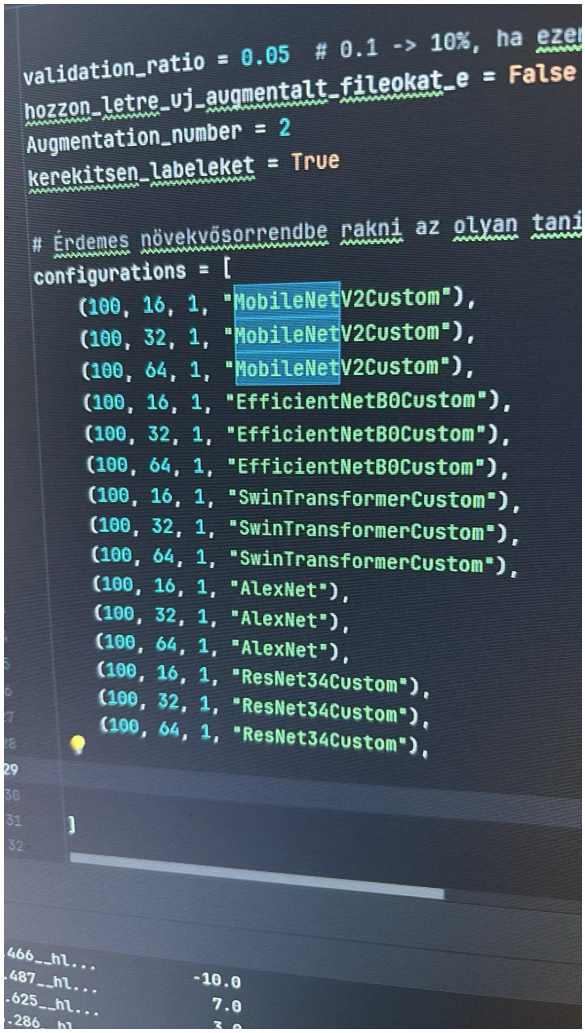

Figure 1: Első konfigurációk

Ekkor még sok problémával küzdöttünk. Ebben a 2 hétben a következőket tanultuk meg.

- **Alexnet** nem lesz alkalmas.
- **Traning loss és Validation loss**-t is érdemes nézni a tanítás során. (és Accuracy-t is)
- **Validálást** a tanító halmazból kell külön választani (5% - 10% - 20%)
- **MobilnetV2** gyorsan tanul és egész jó eredményeket ad. (Utólag kiderült előadáson, miért is...)
- Megtanultuk használni az **LR** -t és elkezdtük monitorozni (lr=0.001 -ről indult általában, finom csökkentéstől jobban tanul) Feleztük, szinusz, koszinusz, logaritmusosan... stb. Majd custom módon.
- **Augmentáció**-val próbáltuk javítani az eredményeket (forgatás, tükrözés), kicsit segít
- **round(label)** , jobban rátanul **tolerance = 0.4 - 0,49**
- **8-16-32 batch size** adja a legjobb eredményeket (minél kisebb, annál pontosabb lehet)
- **Early stop** hasznos (eleinte Traning-loss, majd RMSE alapján)
- Ez nem klasszifikációs feladat..... (Ezt tudtuk, csak Chat-gpt -nek elvitte a gondolatmenetünket egy rossz irányba...)
- Nincs olyan kép, ami nem illik a többi közé, azonban néhanynak szinte üres a mask-ja.

Hiába minden igyekezet, a legjobb eredmény, hogy sikerült rátanítani a "Dummy Solution" -ra. Két hét után ott álltunk, hogy mindent kipróbáltunk, de a lista alján állunk. Úgy gondoltuk, hogy több számítási kapacitásra van szükségünk, ebben a sebességben nem lesz meg az aláírás, ha 3 óra 1 tanítás. Be is szereztünk barátoktól 1-1 erős gépet, és pár apróbbat finomhangolásokat tesztelni. (összesen 5 gépből ált a flotta, de a végén csak a 2 erős gépet használtuk).

Odamentem Samu-hoz (2. helyezett), hogy adjon pár tippet, mert teljesen elvesztünk. A következő tanácsokat adta : Ez egy Regressziós feladat.... Ők még nem is augmentáltak és úgy 1.2 pontot kaptak. ResNet50-et próbáljuk ki. És vegyük ki a ".mask" -okat. Majd RMSE alapján kezdjük el nézni az eredményt. (Ezek közűl a ".mask" -ok kivétele hozta a legtöbb pontot) Hosszas tesztelések után, arra jutottunk, hogy az adatfeldolgozással van a baj.
Ezért megszületett 2. és 3. verzió. Andris csinálta a 2.-at, amíg Gergő a 3.-on dolgozott.

2. verzió :

UPDATES :

- Egyszerűbb kód
- Saját NET
- Dropout + LeakyReLU
- Csak egy kimenet. (regresszió, klasszifikáció helyett.)

Lényegesen egyszerűbb kód, egy óraihoz hasonló netet használt, viszont 1. verzió hibás adat beolvasását használta.

```
configurations = [  # num_epochs, batch_size, dropout, activation
    (2000, 16, 0.2, "ReLU"),  # MSE 16.7519
    (2000, 8, 0.0, "ReLU"),  # MSE 19,5
    (2000, 8, 0.2, "ReLU"),  # MSE 18,5  --> meg kell nezni nagyobb pateint-el
    (2000, 8, 0.1, "LeakyReLU"),  # 20% - 60%  MSE 20
    (2000, 8, 0.2, "LeakyReLU"),  # 20-30, 20-46, 21-49
```

Figure 2: MSE 19,5 = Validation Acc

A betanított netekben való csalódás után ez a NET elkezdett egész jól működni, a dropout-nak és egyszerűségének köszönhetően. Csak írtó lassú volt. Ekkor is használtunk early-stop -ot. Elkezdtem játszani az aktivációs fügvénnyekkel, hátha ez a megoldás. Kis előre lépést jelentett, de nem hozta a hozzá fűzött reményeket.

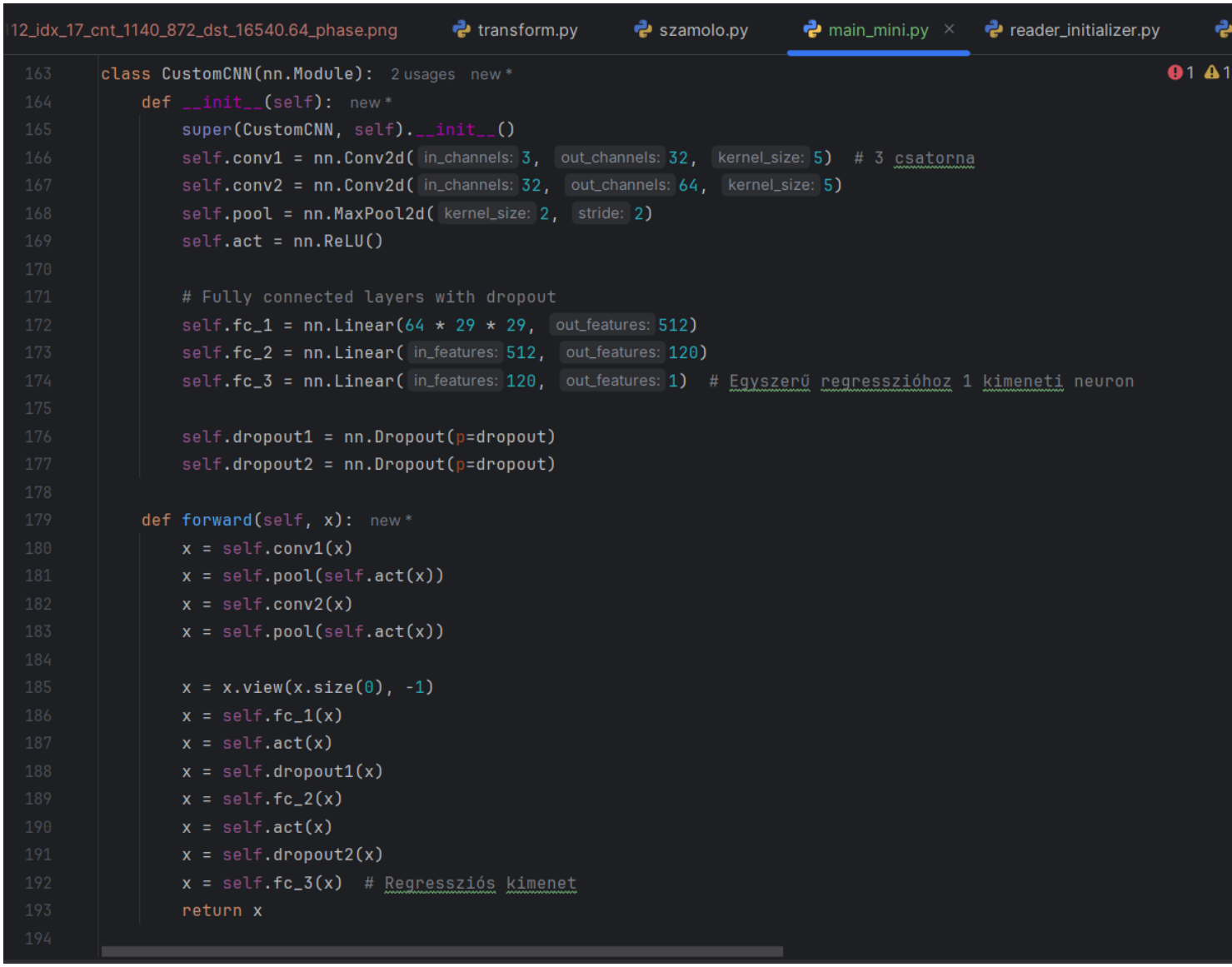

Figure 3: 2. verziós net

Ennél se több se kevesebb réteg nem elég. De még ez se volt elég jó. Ekkor Gergő elkészült a 3. verzióval, ami hiába volt kezdetleges, komolyabb finomhangolás nélkül is eredményesebb volt, mint az első kettő verzió.

3. verzió :

UPDATES :

- OOP kód (Jelentősen megkönnyítette a finomhangolásokat ,lényegesen olvashatóbb kód.)
- új adatbeolvasás, feldolgozás (combined = Image.merge("RGB", (amplitude, phase, amplitude)))
- új NET - Resnet18 / Resnet50
- átgondolt számítási módok
- Tanulási adatok grafikus ábrázolása (plot)

Gyorasn összedobtuk, az első mérést és feltöltöttük. "Score: 1.865" .... Ez már lényeges előrelépés volt. (Eddig a max score: 3 volt) Ekkor a plot volt az ami a következőekben a legnagyobb segítséget nyújtott a hyperparaméterek finomítására. Utána néztünk, hogy a következőt kéne rajta látnunk :

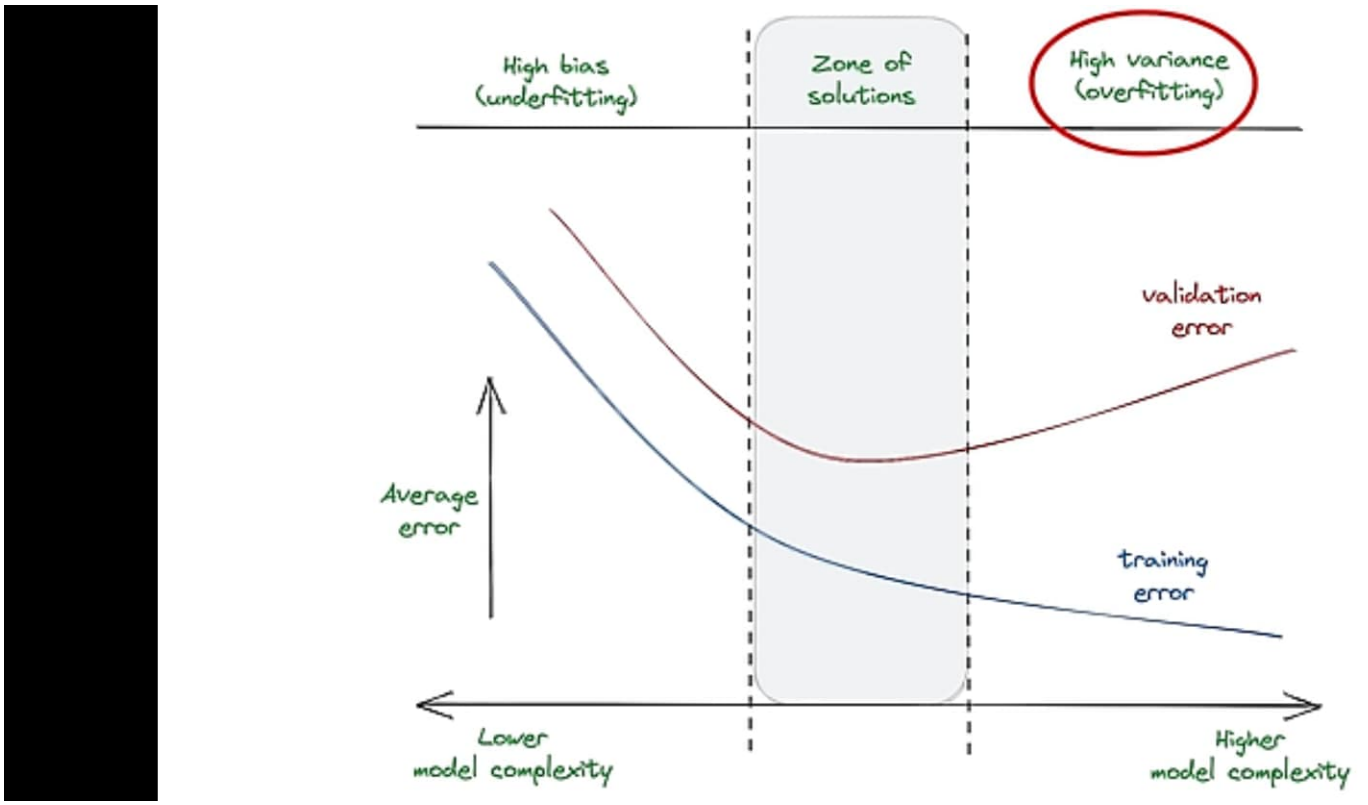

Figure 4: Túltanulás

Elkezdtünk játszani a tanítás ütemével és idejével. Ami a legeredményesebb volt, hogy 0.01 -ről indult az LR, majd ezt felezte 5-8 epochs-onként. Ezekekkel a változtatásokkal a következő eredményt kaptuk :

```
Epoch 35/100, Training Loss: 0.7481, Training Accuracy: 33.80%, Validation Loss: 1.0176, Validation Accuracy: 23.33%, MAE: 0.9853, RMSE: 1.2192, LR: 0.000400
Epoch 36/100, Training Loss: 0.7812, Training Accuracy: 33.33%, Validation Loss: 0.8752, Validation Accuracy: 28.33%, MAE: 0.8669, RMSE: 1.1355, LR: 0.000400
Epoch 37/100, Training Loss: 0.7356, Training Accuracy: 35.48%, Validation Loss: 0.9599, Validation Accuracy: 30.83%, MAE: 0.9345, RMSE: 1.2008, LR: 0.000400
Epoch 38/100, Training Loss: 0.7294, Training Accuracy: 38.84%, Validation Loss: 0.8809, Validation Accuracy: 32.50%, MAE: 0.8621, RMSE: 1.1143, LR: 0.000400
Epoch 39/100, Training Loss: 0.7125, Training Accuracy: 36.97%, Validation Loss: 0.8813, Validation Accuracy: 32.50%, MAE: 0.8651, RMSE: 1.0928, LR: 0.000400
Epoch 40/100, Training Loss: 0.7106, Training Accuracy: 36.23%, Validation Loss: 0.8832, Validation Accuracy: 25.83%, MAE: 0.8597, RMSE: 1.1143, LR: 0.000400
Epoch 41/100, Training Loss: 0.6817, Training Accuracy: 38.94%, Validation Loss: 0.8321, Validation Accuracy: 31.67%, MAE: 0.7933, RMSE: 1.0465, LR: 0.000160
Epoch 42/100, Training Loss: 0.6549, Training Accuracy: 40.34%, Validation Loss: 0.8396, Validation Accuracy: 30.83%, MAE: 0.8107, RMSE: 1.0550, LR: 0.000160
Epoch 43/100, Training Loss: 0.6506, Training Accuracy: 42.86%, Validation Loss: 0.8308, Validation Accuracy: 37.50%, MAE: 0.8193, RMSE: 1.0866, LR: 0.000160
Epoch 44/100, Training Loss: 0.6459, Training Accuracy: 40.90%, Validation Loss: 0.8728, Validation Accuracy: 30.83%, MAE: 0.8558, RMSE: 1.0987, LR: 0.000160
Epoch 45/100, Training Loss: 0.6337, Training Accuracy: 42.39%, Validation Loss: 0.7633, Validation Accuracy: 40.83%, MAE: 0.7365, RMSE: 1.0048, LR: 0.000160
Epoch 46/100, Training Loss: 0.6400, Training Accuracy: 42.67%, Validation Loss: 0.8080, Validation Accuracy: 28.33%, MAE: 0.7915, RMSE: 1.0302, LR: 0.000160
Epoch 47/100, Training Loss: 0.6347, Training Accuracy: 40.62%, Validation Loss: 0.9106, Validation Accuracy: 30.00%, MAE: 0.8986, RMSE: 1.1548, LR: 0.000160
Epoch 48/100, Training Loss: 0.6163, Training Accuracy: 42.39%, Validation Loss: 0.8751, Validation Accuracy: 31.67%, MAE: 0.8543, RMSE: 1.1207, LR: 0.000160
Epoch 49/100, Training Loss: 0.6093, Training Accuracy: 44.63%, Validation Loss: 0.8347, Validation Accuracy: 39.17%, MAE: 0.7891, RMSE: 1.0658, LR: 0.000160
Epoch 50/100, Training Loss: 0.6141, Training Accuracy: 44.72%, Validation Loss: 0.7890, Validation Accuracy: 44.17%, MAE: 0.7761, RMSE: 1.1189, LR: 0.000160
Epoch 51/100, Training Loss: 0.5798, Training Accuracy: 47.34%, Validation Loss: 0.8109, Validation Accuracy: 35.83%, MAE: 0.7914, RMSE: 1.0615, LR: 0.000160
Epoch 52/100, Training Loss: 0.6018, Training Accuracy: 42.95%, Validation Loss: 0.8735, Validation Accuracy: 34.17%, MAE: 0.8470, RMSE: 1.1433, LR: 0.000064
Epoch 53/100, Training Loss: 0.5757, Training Accuracy: 46.13%, Validation Loss: 0.8906, Validation Accuracy: 30.83%, MAE: 0.8766, RMSE: 1.1804, LR: 0.000064
Epoch 54/100, Training Loss: 0.5619, Training Accuracy: 48.37%, Validation Loss: 0.8706, Validation Accuracy: 35.00%, MAE: 0.8663, RMSE: 1.1578, LR: 0.000064
Early stopping
```

Figure 5: Aznap esti eredmények

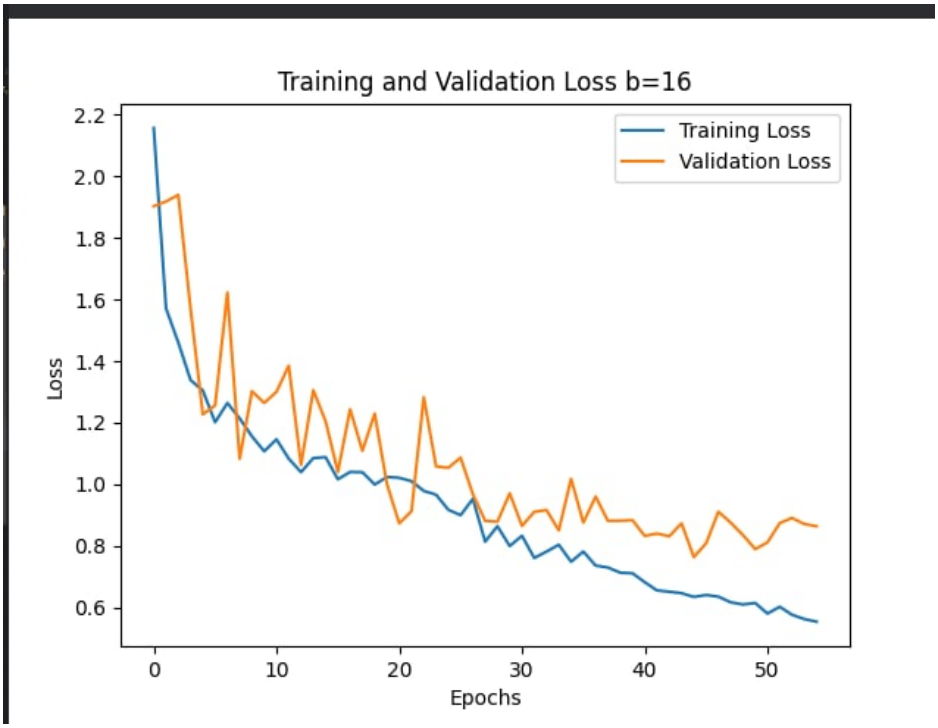

Figure 6: PLOT

47 és 49. -ik epoch- ot töltöttük fel. Előbbi 1.170 ponotot, útóbbi 1.112 -t kapott. Ezzel Samuékkal holtversenyben első helyre kerültünk. A kód egyszerűsége volt ekkor a fegyverünk és persze az elözö 2 verzióban szerzett tapasztalati tudás. Ekkor még a learning rate -et úgy csökkentettük, hogyha nem tanul adott ideig, akkor LR az elözö LR 40% -a lesz. Ahogy néztem ez egy elég egyszerű módszer a csökkentésére. Ennél csak bonyolultabbak vannak.

Majd ezt úgy finomítottuk tovább, hogy ehhez a verzióhoz Gergő készített egy külön osztályt, amely a tanulási adatokból próbálja kiszámolni az lehető legoptimálisabb learning ratet, ezzel produkálta a legjobb elért pontosságot a neurális háló. (LRAdjuster.py)

Működése:
patiance: hány epoch után csökkenti a learning ratet,
factor: mekkora ütemben csökkentse a leraning ratet.
tolerance: mekkora veszteséget tekintünk javulásnak.

Majd utolsó napokban arra jöttünk rá, hogy nem fixáltuk a random felosztásokat, ez azt jelentette, hogy minden futtatásra eltérő eredményt adott a NET. (Mivel mindig máshogy osztotta fel a validációs és betanító adathalmazt és a betanításkor más sorrendbe kapta meg a hálózat az adatokat.) Ezeket a különböző eredményeket értékeltük ki.

2 Végső kód :

Figure 7:

- **LRAadjuster.py**:

- **dataset.py**: Ez a file volt felelős a tanító és kiértékelési halmaz tárolásáért és feldolgozásáért. Test és Train képek itt kerültek feldolgozásra, kicsit különböző módon.

- **main.py**: A fő futtatási szkript, amely összefogja az összes többi modult. Innen indul az egész folyamat, például az adatok betöltése, a modell edzése és az eredmények kiértékelése. Ez a kód fogta össze az összes többit. Implementálta az adathalmazokat. Meghívta a beolvasást, majd a tanítást. Végül az eredményeket kirajzolta plot segítségével.

- **metrics_utils.py**: Tanulási statisztikákat számolja.

- **model.py**: A fájl tartalmazza a neurális hálózat architektúráját és a modell definiálásához szükséges komponenseket.

- **out_csv.py**: Az eredmények és más fontos adatok CSV formátumban történő exportálásáért felelős modul. Ezt a "trainer.py" hívta meg, vagy a tanítás közben, ha megfelelően pontos volt és a tanítás végén, amikor "Early-stop"-olt.

- **trainer.py**: Egy példány ami, a modell betanításának logikáját tartalmazza, például az edzési lépéseket, visszacsatolást (backpropagation), valamint a validációs és tesztelési folyamatokat. Majd ez hívja meg a **out_csv.py**, ha megfelő pontosságot ért el.