Mélytanulás projektfeladat

készítette: **Balla Krisztián**
bkrisz31@gmail.com
neptun-kód: **RZWVC0**
ágazat: **Adattudomány és mesterséges intelligencia**

Point cloud segmentation with graph neural networks

Feladat:
The goal of this project is to delve into point cloud segmentation using Graph Neural Networks (GNNs). You have to work with the ShapeNet dataset, which contains multiple object categories (from airplanes to wine bottles). By choosing this project, you'll gain expertise in deep learning and spatial data analysis.

LLM használat
A projektmunka során az alábbi feladatokhoz használtam LLM-et:

- Forráskódhoz komment generálás
- Vizualizációs programkód
- Irodalomkutatás, State-of-The-Art megoldások áttekintése
- Docker megoldások, hogy egyetlen build parancs után futtatható legyen a kód

Tanév: 2024/25. tanév, I. félév

1. Bevezetés

A projekt során felhasznált irodalmak, illetve tartalmuk rövid leírása alább található:

- **PointNet++** [1]: Alapvetően elődjét, a PointNet-et [2] fejlesztették tovább. Hierarchikus megközelítést vezet be a pontfelhők feldolgozására, mely lehetővé teszi, hogy a hálózat lokális mintázatokat tanuljon meg különböző skálákon, hasonlóan a konvolúciós nurális hálózatokhoz (CNN-ek) a képek esetében. Az egyes pontok körül lokális tulajdonságokat vizsgál, valamely csoportosítás, illetve mintavételezés segítségével.
- **SPoTr** [3]: Ellentétben a PointNet architektúrával, a SPoTr egy 'figyelem' alapú neurális háló, ami a transzformerekre épít. Egyik fő kontribúciója, az "Efficient Global Attention", amely a pontfelhővel dolgozó transzformer alapú hálók egyik nagy hátrányát, a nagy számítási kapacitás igényt oldja meg, úgy, hogy úgynevezett 'Self Positioning (SP)' pontokat hoznak létre, melyek adaptívan helyezkednek el a bemeneti pontfelhő alapján. A másik fő eredményük az úgynevezett 'Disentangled Attention', amely mind a térbeli, mind a szemantikus kapscolatokat figyelembe veszi a pontok között, ezzel jelentősen növeli a komplex geometrikai és szemantikai kifejezőképességét.
- **PointNeXt** [4]: Bár számos érdekes újdonságot mutattak be, én egy framewrok-öt, az OpenPoints-ot [1] vettem át tőlük, mely a pontfelhők feldolgozásával foglalkozó neurális hálóknak biztosít egy egységes, és igazságosan összehasonlítható környezetet.

2. Adatszett

A munkám során használt adatszett a ShapeNetPart [2], mely 16 különböző osztályú objektumról tartalmaz annotált pontfelhőket, ahol az objektumok az ősszetartozó részeik szerint vannak annotálva (repülőgép szárnya, teste, hajtúműve stb.). Az adatszettben közel 17000 pontfelhő található, és az egyik legelterjedtebben használt pontfelhő szegmentálási feladatokra. Az adatszetter 80-10-10 arányban osztottam tanító, validáló és tesztelő részekre, melyhez természetesen saját adataszett osztály is készült.

3. A neurális hálók és tanításuk

3.1. Baseline Model

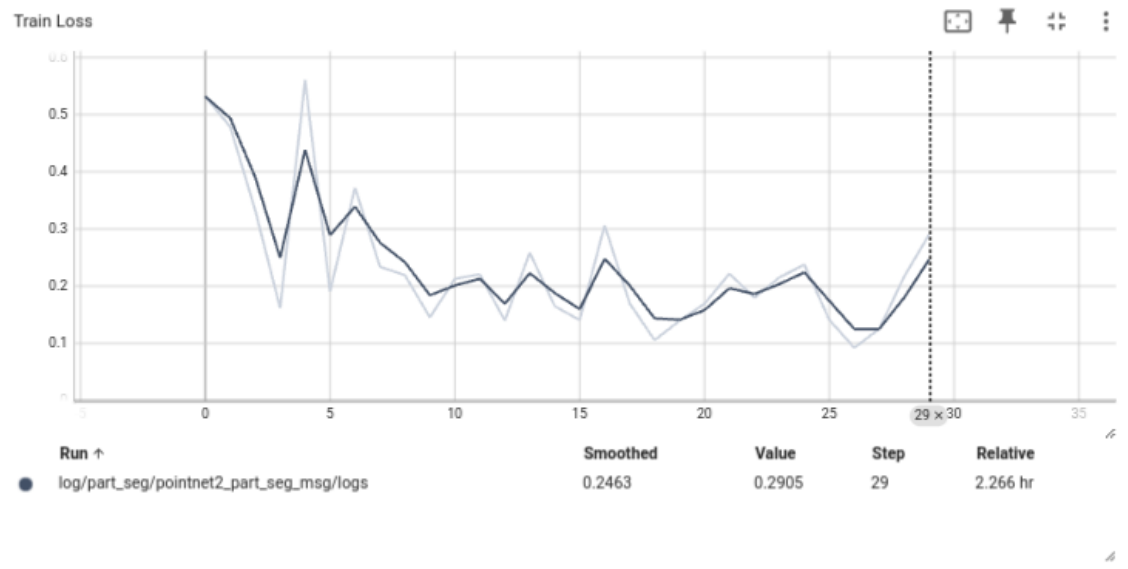
Részben a feladat félreértése miatt, részben azért, mert transzformer alapú neurális jálóval szerettem volna dolgozni, a Milestone II beadására én a PointNet++ Neurális hálót tanítottam fel az általam készített adatosztály segítségével a ShapeNetPart adatszetten. Mivel ez csak egy alap modellnek szolgál, így itt nem történt hiperparaméter optimalizálás. A hálót tanítását 30 epoch után felfüggesztettem, ugyanis az aktuális konfigurációval elértem a modell teljesítőképességének határát. A tanítás az alábbi hiperparaméterekkel történt.

Hyperparameter	**Value**
Learning Rate (η)	0.001
Batch Size	32
Epochs	29
Momentum	0.9
Weight Decay	0.0004
Dropout Rate	0.2
Optimizer	Adam
Activation Function	ReLU
Learning Rate Schedule	Step decay

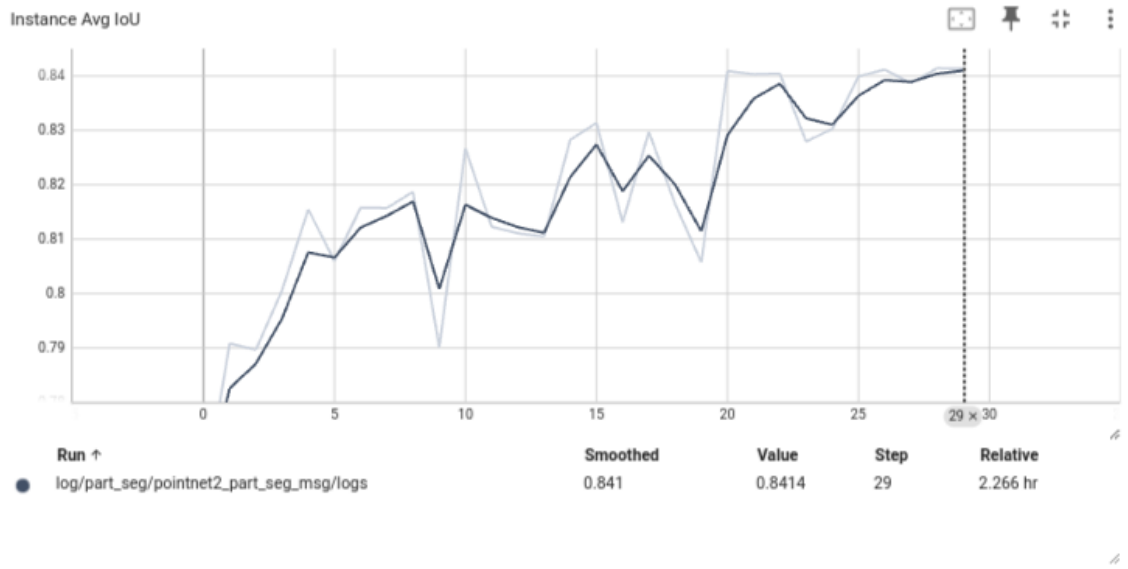
1. táblázat. PointNet++ training hyperparameters

[1] https://guochengqian.github.io/PointNeXt/
[2] https://shapenet.org/download/shapenetsem

A legfontosabb metrikák a tanítás során az 1. és 2.ábrákon láthatók.



1. ábra. A loss alakulása a PoitNet++ tanítás során



2. ábra. Az objektumszintú iou alakulása a PoitNet++ tanítás során

3.2. Az általam tervezett neurális háló

A háló tervezesénél a transzformer alapú megoldásokon kívül a PoinNet családban bemutatott módszereket is figyelembe vettem. Az általam tervezett hálónak 3 fő része van, melyek az alábbiak:

- **Encoder**: Az encoder alapját a PointVit [5] transzformer alapú neurális háló ihlette. A pontfelhőt csoportosítom a P3Embed módszer segítségével, amely inkrementálisan kisebb csoportokra bontja a pontfelhőt, így a jellemzők különböző szinteken is rendelkezésre állnak majd a decoder számára. A pontok relatív (egymáshoz) viszonyított pozíciója alpján történik a csoportosítás. Ezek a csoportok szolgálnak tokenként a későbbi rétegekben, melyek között az encoderben található Multi-Head

Attention rétegek segítségével globális összefüggéseket állapítok meg. Az encoder kimenete a pontfelhő különböző skálázási szintjeinek megfelelő tokenjeihez készült globális (és a P3Embed miatt lokális) figyelem mátrixok, mind a koordinátákhoz, mind a jellemzőkhöz (felületi normálisok).

- **Decoder**: A decoder architekrúra lépésekben dolgozza fel az encoder kimenteit. Először globális konvolúciókat használ az enkódolás során készített jellemző csoportok dekódolásához, melyeket egy one-hot enkódolt klasszifikáló token készítéséhez használ fel. Ennek a tokennek a segítségével dekódolja a koordináta csoportokat (visszafele haladva), melyhez a PointNet++-ban ismertetett feature propagationt és a SPoTr-ban bemutatott LPAMLP (Local Point Attention + Multi Layer Perceptron) blokkot használja.
- **Classification Head**: A classification head lokális és globális jellemzőket nyer ki a dekódolt jellemzőkből és koordinátákból Fully Connected rétegek segítségével, és ezeket összefűzi egy jellemző mátrixba. Ezen a mátrixon szemantikus és geometriai jellemzőket nyer ki egy Multi-Head Attention réteg segítségével, melyhez hozzáadja az a classification head bemenetét is egy residual blokk segítségével, hogy elkerüljem, hogy eltűnjön a gradiens már rögtön a backpropagation elején. A végső osztály predikciókat pedig egy 1D-s konvolúciós réteg rendeli hozzá a bemeneti pontokhoz.

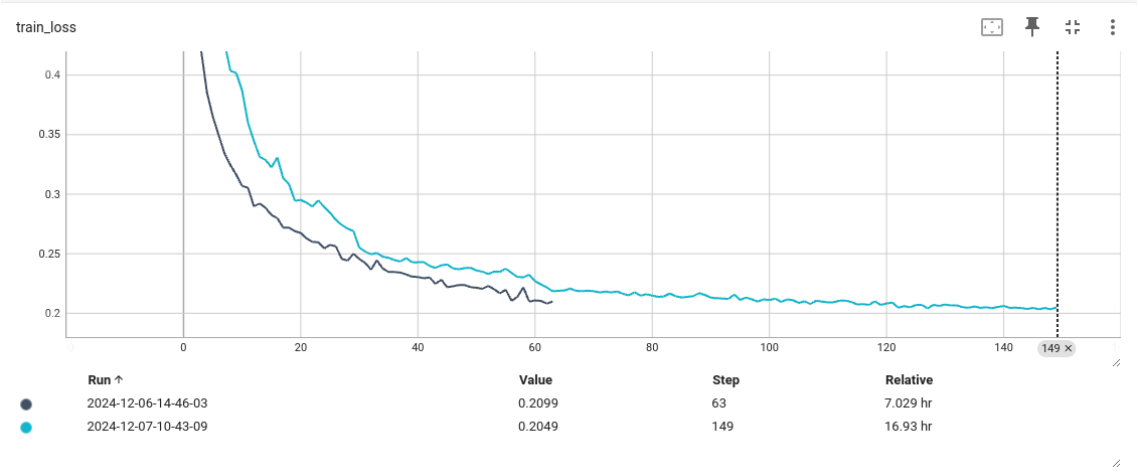
Mint a bevezetésben ismertettem, az implementációhoz az OpenPoints frameworkot használtam fel, melyet a terjedelemre, illetve szükségességre való tekintettel itt nem részletezek.

Mivel a hálónak közel 90 millió paramétere van, így az automatikus hiperparaméter hangolás rengeteg időt vett volna igénybe, így manuálisan hangoltam. A két legjobban teljesítő tanításnak a következő hiperparaméterei voltak.

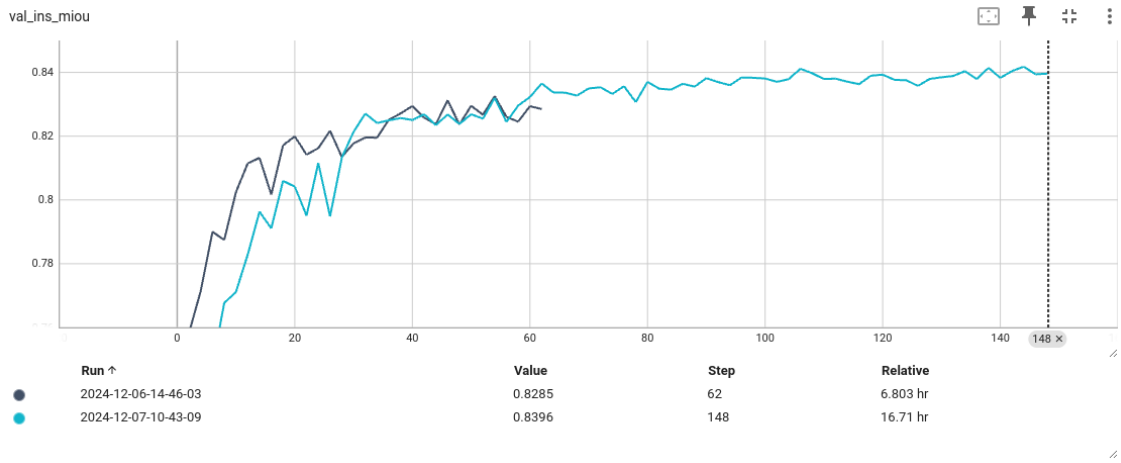
Hyperparameter	**Run 1**	**Run 2**
Learning Rate (η)	0.004	0.001
Batch Size	12	12
Epochs	62	148
Weight Decay	0.001	0.004
Dropout Rate	0.0	0.1
Optimizer	Adam	AdamW
Activation Function	ReLU	LeakyReLU
Learning Rate Schedule	Step decay	MultiStep
Augmentation	None	Jitter, Scaling, Rotation

2. táblázat. Hyperparameters for the two best performing runs

A tanítások során a loss (3.ábra) és az objektumszintű miou (4.ábra) metrikák alább láthatók.



3. ábra. A loss alakulása a saját hálóm tanítása során



4. ábra. Az objektumszintú iou alakulása a saját háló tanítása során

4. A hálók kiértékelés, felhasználói interfész

Mind a PointNet++, mint a saját hálóm a ShapeNetPart adatszett tanító részén lett kirtékelve, melynek eredménye alább látható.

Class	My Model	PointNet++
Airplane	0.782713	0.821437
Bag	0.622064	0.735315
Cap	0.808636	0.841750
Car	0.696074	0.737776
Chair	0.881143	0.905016
Earphone	0.708238	0.714715
Guitar	0.892955	0.904174
Knife	0.804441	0.877231
Lamp	0.805346	0.842915
Laptop	0.950196	0.946049
Motorbike	0.537377	0.595754
Mug	0.881492	0.948060
Pistol	0.760589	0.798120
Rocket	0.423337	0.575504
Skateboard	0.703905	0.751842
Table	0.807279	0.820963

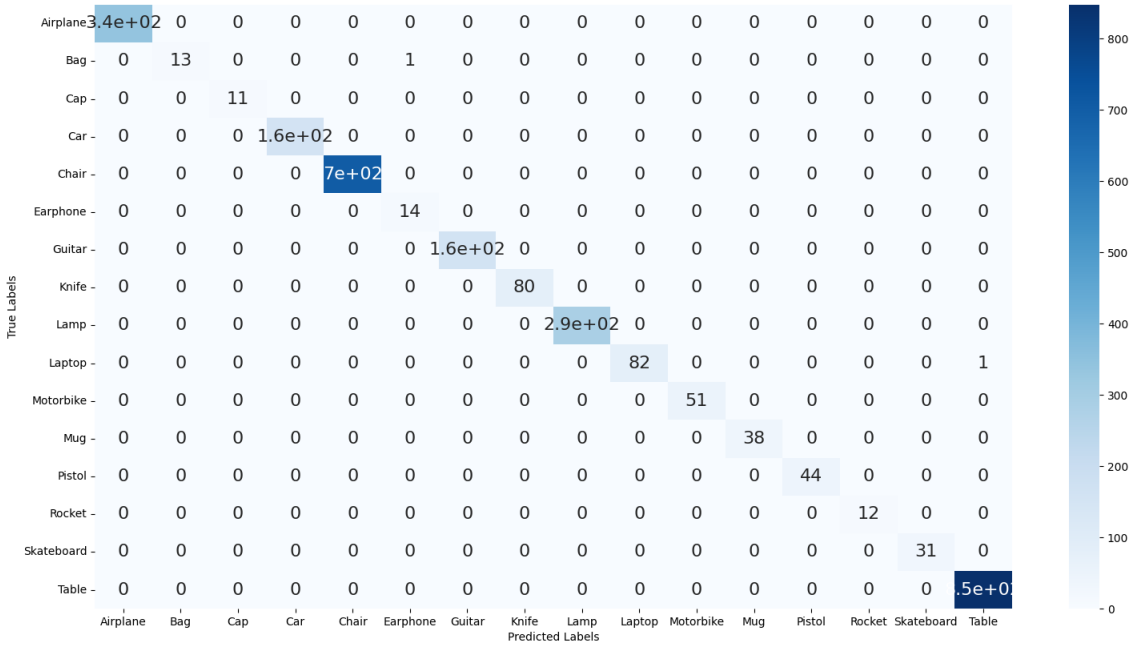
3. táblázat. Class-wise mIoU Scores on the test dataset

Metric	My Model	PointNet++)
Accuracy	0.92543	0.93935
Class avg accuracy	0.79973	0.83077
Class avg mIoU	0.75842	0.80104
Instance avg mIoU	0.81770	0.84371

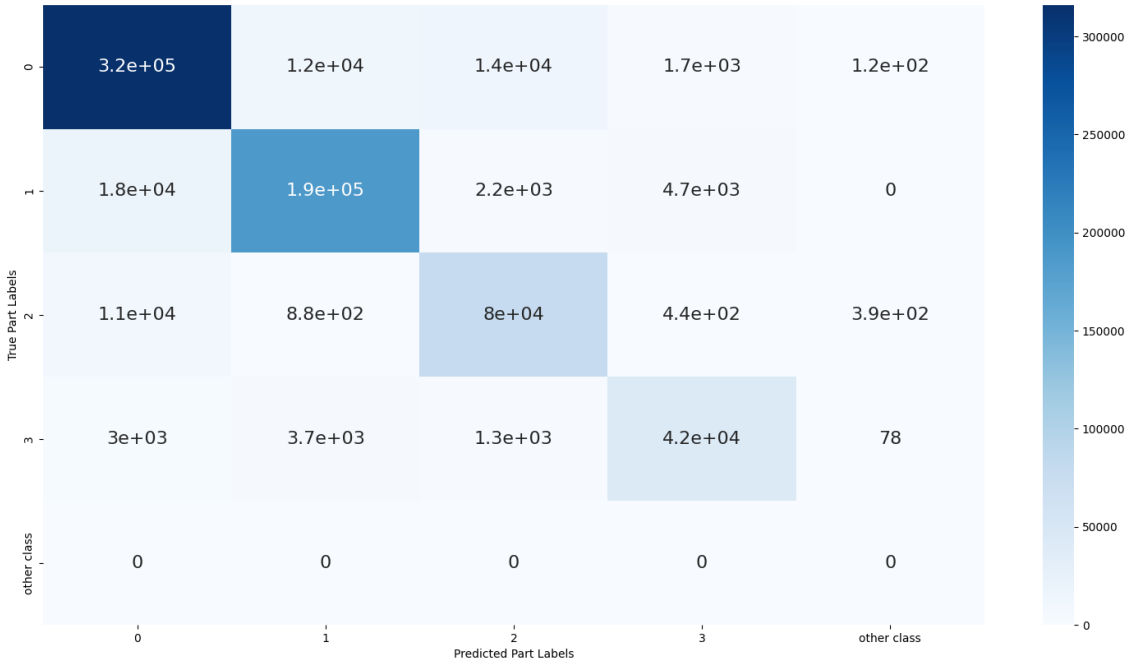
4. táblázat. General Metrics on the test dataset

Mivel ez egy Multi-Class azon belül pedig egy multi part klasszifikációs feladat, így az alábbi két tévesztési mátrixot készítettem hozzá. Az elsőn (5.ábra) az osztály szintű klasszifikáció látható, ahol az adott pontfelhő prediktált osztályát a prediktált objektumon belüli 'part'-ok leggyakrabban előfordulú értéke

határozta meg (az objektum szintű 'part' címkék 0..49 vannak). A 6 ábrán pedig a repülőgép osztályhoz tartozó objektumon belüli 'part' címkék tévesztési mátrixa látható.



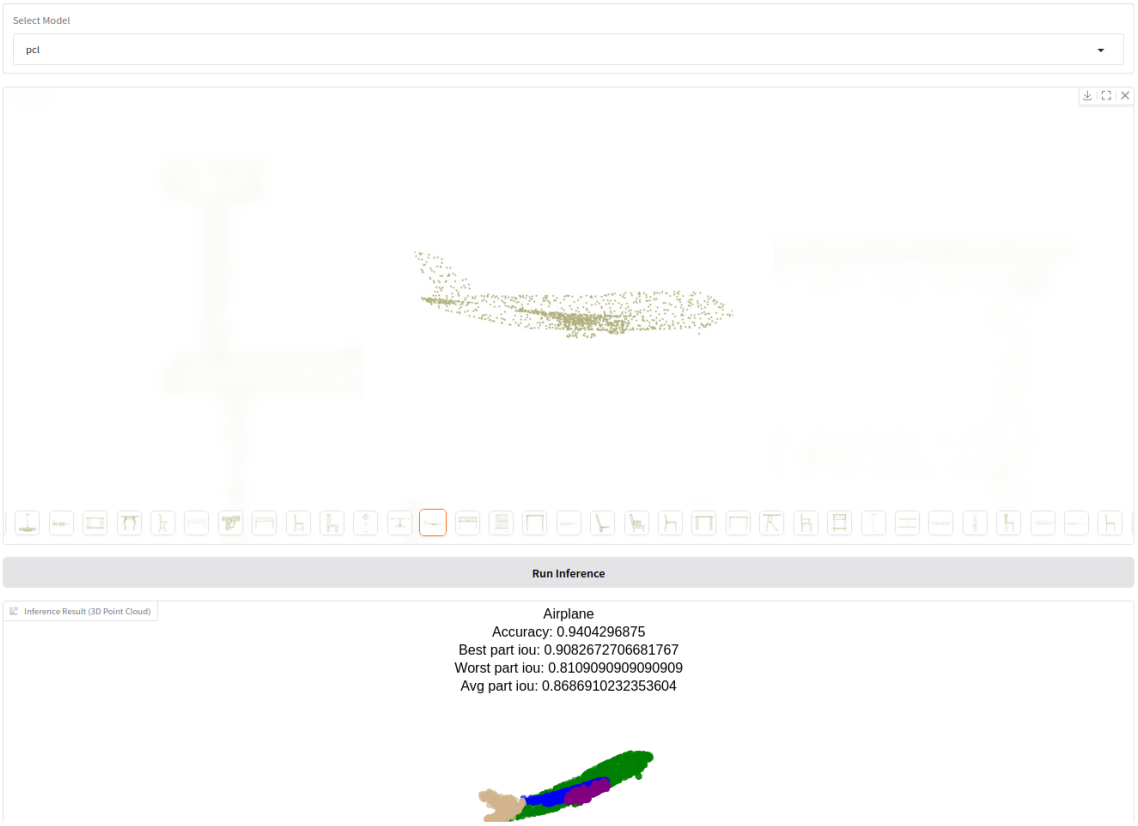
5. ábra. Osztály szintű tévesztési mátrix



6. ábra. A repülőgép osztály szegmentálásnak tévesztési mátrixa.

Ezen felül gradio segítségével egy user interfész is készült, melyen a böngészőben ki lehet választani, hogy melyik modellt szeretné a felhasználó használni (bármikor módosítható). Ezen felül 50 darab pontfelhőhöz 2D-s vetített képet is készítettem, melyek közül úgy lehet válogatni, mint a telefon galériájában. Amennyiben kiválasztotta a user a neki szimpatikus 'pontfelhőt' (képet) a 'Run Inference' gombra kattintva lefut a kiválasztott hálónak a forward metódusa és megjeleníti a 3D-s, szegmentált pontfelhőt, amely tetszés

szerint mozgatható, forgatható. A gradio-s user interfészt szemlélteti a 7.ábra.



7. ábra. Felhasználó interfész model inferenciához

5. Konklúzió

Bár előzetes várakozásaimnak megfelelően nem sikerült egy jobban teljesítő modellt tervezi a PointNet++-nál, az általam tervezett háló így is jól teljesített. Véleményem szerint, ha több adat (a transzformerek különösen profitálnak belőle), illetve több idő állt volna rendelkezésemre hiperparaméter optimalizálása (egy teljes tanítás közel 16 óráig tartott), akkor a tervezett háló képes lenne túlteljesíteni a PointNet++-t.

A tanulmányozott irodalom jegyzéke

[1] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas, *PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space*, 31st Conference on Neural Information Processing Systems (NeurIPS), 2017

[2] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas, *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017

[3] Jinyoung Park, Sanghyeok Lee, Sihyeon Kim (Korea University), Yunyang Xiong (Meta Reality Labs), and Hyunwoo J. Kim (Korea University), *Self-positioning Point-based Transformer for Point Cloud Understanding*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2023

[4] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Hammoud, Mohamed Elhoseiny, and Bernard Ghanem, *PointNeXt: Revisiting PointNet++ with Improved Training and Scaling Strategies*, Conference on Neural Information Processing Systems (NeurIPS), 2022

[5] uxin Wang, Yueqi Du, Yifan Xu, Cheng Zhang, *PointViT: Efficient Vision Transformers for Point Cloud Recognition*, EEE/CVF International Conference on Computer Vision (ICCV), 2021