

DIPLOMADOLGOZAT

Bagi Gergő
Gépészmérnök

Gödöllő
2025



Magyar Agrár- és Élettudományi Egyetem
Szent István Campus
Gépészmérnöki Szak
Műszaki fejlesztő specializáció

Mesterséges Intelligenciával
Támogatott Anyagvizsgálat

Belső konzulens: Dr. Tóth János
Egyetemi adjunktus
Külső konzulens: Csizmadi Dániel
Szakcsoportvezető

Készítette: **Bagi Gergő**
DZSJ1K
Nappali tagozat
Intézet/Tanszék: **Műszaki Intézet/**
Mechatronika Tanszék

Gödöllő
2025

Tartalomjegyzék

1.	Bevezetés.....	5
1.1	Téma jelentősége.....	5
1.2	Célkitűzés.....	6
2.	Szakirodalom feldolgozása	7
2.1	A minőség vizsgálata	7
2.2	Anyagszerkezeti vizsgálat – Metallográfia	8
2.3	BGA (Ball Grid Array - golyós rács tömb)	10
2.4	A mesterséges intelligencia.....	11
2.5	Objektum felismerés konvolúciós neurális hálóval (CNN).....	11
2.5.1	A YOLO modell (You Only Look Once).....	13
2.5.2	YOLO Loss Function – Veszteségfüggvény	15
3.	Anyag és módszer	17
3.1	Csiszolatkészítés jelentősége	17
3.2	Csiszolatkészítés menete.....	18
3.3	Program.....	20
3.3.1	Grafikus felhasználói felület (GUI).....	21
3.3.2	A main() függvény.....	21
3.3.3	Az üdvözlő ablak (WelcomePage)	21
3.3.4	SMD és THD ablakok	23
3.3.5	A BGA ablak	24
3.4	A BGA modul működése.....	25
3.4.1	Load Image	25
3.4.2	Load Image Folder.....	26
3.4.3	„Előző” és „következő” navigációs gombok.....	27
3.4.4	Run Analysis.....	27

3.4.5	Auto-Process.....	28
3.4.6	Save Results.....	29
3.5	A mesterséges intelligencia tanítása	32
3.5.1	Címkézés	32
3.5.2	Tanítás	34
3.5.3	Eredmény	35
3.5.4	Értékelés	39
4.	Gazdasági számítás	40
4.1	Összegzés	40
5.	Összefoglalás.....	41
6.	Summary	42
7.	Nyilatkozatok	43
8.	Irodalomjegyzék.....	48
9.	Mellékletek.....	52
9.1	Main.py	52
9.2	BaseMainWindow.py.....	52
9.3	WelcomePage_File.py	52
9.4	SMDWindow_File.py	55
9.5	THDWindow_File.py	56
9.6	BGAWindow_File.py	58
9.7	Az első tanításhoz tartozó képek és diagrammok	70
9.8	A második tanításhoz tartozó képek és diagrammok.....	93

1. Bevezetés

"Quality means doing it right when no one is looking." – mondta Henry Ford amerikai iparos, az autóipar forradalmasítója. A minőség azt jelenti, hogy akkor is jól csináljuk a dolgokat, amikor senki sem figyel. A minőség és annak fogalma nehezen írható le egyetlen definícióval, az ember csak akkor ismeri fel, ha látja. Transzcendens módon elmondható róla, hogy a minőség az, ami kiállta a próbát. A próba pedig jól definiálható tesztek, vizsgálatok formájában, melynek eredménye a minőség vagy annak hiányának megállapítása.

1.1 Téma jelentősége

A minőség és annak igénye visszanyúlik egészen az őskorba, ahol a létfenntartás szükségleteként volt jelen. Az ember érzékszerveivel tapasztalta és ellenőrizte a körülötte lévő világot. Ezt mai értelemben az érzékszervek útján történő minőség-ellenőrzésnek tekinthetjük. A ma alkalmazott minőségirányítási rendszerek alapjai a középkorban megjelent céhektől származnak. A manufaktúrák elterjedésével megnövekedett a termelési volumen, a gyártás és az értékesítés folyamata elkülönült. A vásárlókban felmerült az igény az iránt, hogy a megvásárolt termék minősége garantált legyen. Ennek teljesítésére a céhek előre rögzítették az alapanyag minőségét, a gyártási technológiát és folyamatosan ellenőrizték, hogy a késztermék megfelel-e az elvárt minőségnek. A minőség iránti igény tovább erősödött. A minőség-ellenőrzés új szintje a minőség-szabályozás lett majd a minőségbiztosítás és így jutottunk el a napjainkban is működő Teljes körű minőség-irányítási rendszerekhez (Total Quality Management).

A dolgozatomban elektronikai alkatrészek forrasztásának minőségi ellenőrzését mesterséges intelligencia modellel végeztetem. A mesterséges intelligencia fékezhetetlenül tör be az élet minden szegmensébe, egyre több és egyre komplexebb feladatok bízhatóak rájuk, a technológia fejlődése pedig megköveteli a jövő szakembereitől, hogy ismerjék és hatékonyan használják. A mesterséges intelligencia jól alkalmazható osztályozásra, klaszterezésre, képen szereplő objektumok tulajdonságainak érzékelésére, ebből következően alkalmas képfeldolgozási és kiértékelési feladatok automatizálására. Az automatizálás már valósággá vált a jelenben és az új irányoknak köszönhetően még többet várhatunk a jövőben.

1.2 Célkitűzés

A dolgozatom célja egy mesterséges intelligencia modell feltanítása BGA (Ball Grid Array - golyós rácstömb) lábak keresztcsiszolati képeinek feldolgozására és kiértékelésére. A vizsgálati szempont a forrasztás és az abban esetlegesen kialakuló üres térrész egymásra vonatkoztatott százalékos arányának meghatározása és a minőségi előírás szerinti elbírálása. A felhasználhatóság érdekében a GUI (Graphical User Interface – Grafikus Felhasználói Felület) létrehozása is a dolgozat részét képezi.

2. Szakirodalom feldolgozása

Ebben a fejezetben bemutatásra kerülnek az Anyag és módszer fejezetben foglaltak megértését segítő szakirodalmak feldolgozott formában.

2.1 A minőség vizsgálata

Ebben a fejezetben a kész és félkész termékek, valamint az alapanyagok vizsgálatának lehetőségei olvashatók.

A termékek mechanikai vizsgálata két fő csoportba sorolható, ezek a roncsolásos és roncsolásmentes eljárások. Alkalmazásukat a kívánt eredmény érdekében határozzuk meg. [1]

1. Roncsolásmentes anyagvizsgálati eljárások [2]

A roncsolásmentes vizsgálatok további két részre bonthatóak, felületi hibák vizsgálatára, valamint az anyagok belsejében lévő hibák feltárására irányulóan. [3]

Felületi hibák vizsgálata:

- vizuális megfigyelés
- mágneses repedésvizsgálat
- penetrálófolyadék vizsgálat
- magnetoinduktív vizsgálat

Belső hibák vizsgálata

- röntgen
- ultrahangos vizsgálat

2. Roncsolásos anyagvizsgálati eljárások [4]

A roncsolással járó anyagvizsgálatokat próbatesteken vagy kész termékeken alkalmazzák, utóbbi esetben a termék a vizsgálatot követően jellemzően nem használható fel, selejt.

- szilárdsági vizsgálat
- keménységvizsgálat
- anyagszerkezeti vizsgálat

A dolgozat relevanciájának szempontjából a következőkben az anyagszerkezeti vizsgálat kerül részletesebb bemutatásra. [5]

2.2 Anyagszerkezeti vizsgálat – Metallográfia

A metallográfia az anyagok belső szerkezetének, struktúrájának feltárásával foglalkozik. Célja azok tulajdonságainak vizsgálata és hibaanalízise. [6]

Az eljárás során fémek, ötvözetek, kerámiák, polimerek és más kompozit anyagok is vizsgálhatóak. A vizsgálat során információhoz juthatunk az anyag kristályszerkezetét illetően, megismerhetjük az anyagot alkotó fázisok milyenségét és mennyiségét, meghatározhatjuk a szemcsék nagyságát és azok eloszlását. Feltárhatók az anyagban rejlő hibák, ilyenek a zárványok, repedések inhomogenitások, dúsulások vagy a korrózió. A metallográfia a gyakorlatban minták síkcsiszolatának készítését és mikroszkóp alatti vizsgálatát jelenti. A következőkben a síkcsiszolatok elkészítési módjáról olvashat. [7] [8]

1. Mintavétel

A mintavétel során olyan részt kell választani, amely hordozza az anyagra jellemző tulajdonságokat, anyaghibákat. A kijelölést követően a minta kivágása következik. Fontos, hogy a kivágás eszközét és a vágási paramétereket a mintadarabnak megfelelően válasszuk meg. Az ilyen jellegű vizsgálatok esetében mechanikai- és hőhatás nem érheti a mintadarabot, hiszen ez megghamisítaná az eredményt.

2. Mintabefogás és beágyazás

A mintadarabok befogása több szempontból is fontos lépése a folyamatnak, ennek legelterjedtebb eszköze a műgyanta. A kis méretű minták kézzel történő csiszolása nem lenne kivitelezhető beágyazás nélkül, emellett a gyanta körbe ölelve rögzíti a vizsgálni kívánt részt. Nem utolsó szempont, hogy az automata polírozógépek szabványos átmérőjű mintákkal dolgoznak, valamint az sem, hogy a gyantába ágyazott mintákra könnyen vihető fel jelölés a vizsgálat nyomon követése céljából.

3. Csiszolás

A csiszolás a vizsgálandó rész megközelítését teszi lehetővé, a vizsgálandó síkot készíti elő polírozásra. A csiszolás során a legdurvább csiszolópapírtól indulva halad a folyamat a legfinomabb csiszolópapírig. A csiszolópapírok jellemzője a szemcseméret, számozásuk a szemcsék méretére utal. Minél nagyobb a szám, annál finomabb a szemcseméret. A csiszolópapírok egyre finomabbra való cserélésével a mintán lévő karcok mélysége csökken.

Amennyiben a mintán már csak a legfinomabb csiszolópapír karcai láthatóak, át lehet térni a polírozás folyamatára.

4. Polírozás

Polírozás során a cél a vizsgálandó felület készre munkálása, az eredménye karcmentes tükörsima felület. A polírozás a csiszoláshoz nagyon hasonlóan, több lépcsőben történik. A korong anyaga lehet bársony, filc vagy más finom tapintású anyag. A folyamat eredményességét polírozó anyag is segíti. Ez jellemzően olyan folyadékot jelent, amelyben szilárd anyagszemcsék vannak eloszlatva, megnevezése: szuszpenzió.

5. Maratás

A maratás a polírozott mintafelületnek oldó vagy színező hatású oldattal történő kezelése. Lehetővé teszi az anyag szemcseszerkezetének vizsgálatát, a maratás hatására a kristályok határvonalai éles vonalként jelennek meg a mikroszkopikus képen.

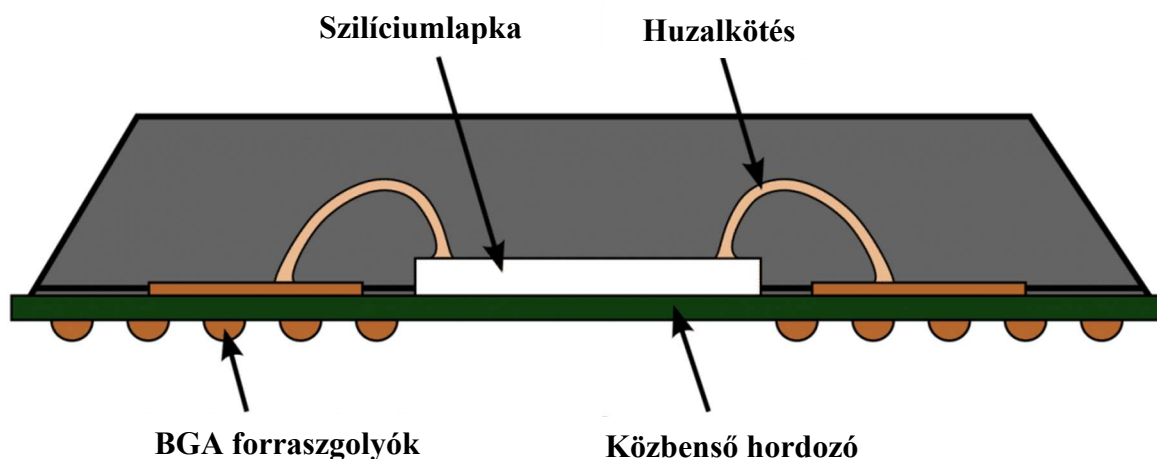
6. Mikroszkópia

A kész csiszolatok vizsgálatára mikroszkópokat használnak. Ezek a visszavert fényt használják a képalkotáshoz, a fénymikroszkópok ezen fajtáit metallográf mikroszkópoknak is nevezik. Megkülönböztetünk normál és inverz mikroszkópokat. Szembetűnő különbség a tárgyasztal elhelyezkedése. Normál esetben a mintadarab az objektív alatt, amíg inverz mikroszkópok esetében a minta az objektív felett helyezkedik el fejjel lefelé fordítva. Ennek előnye, hogy a vizsgált sík merőleges az objektívre, a minta alja nem kell, hogy párhuzamos legyen a tetejével. További előnye, hogy a fókusz állítása során, ha az objektív a mintadarabba ütközik megemeli azt, nem sérül és nem törik el. [9] [10]

2.3 BGA (Ball Grid Array - golyós rácstömb)

A BGA (Ball Grid Array) egy felületszerelt tokozási technológia, amelyet integrált áramkörök (IC-k), például mikroprocesszorok nyomtatott áramköri lapra (PCB) történő csatlakoztatására használnak. A BGA tokozás lényege, hogy a chip alsó oldalán egy rácsszerű elrendezésben apró forraszgolyók (solder balls) találhatók, amelyek a PCB megfelelő padjeihez csatlakoznak.

A hagyományos lábkivezetéssel rendelkező tokozási megoldásokkal szemben nagy előnye, hogy kialakításából adódóan nagyobb számú csatlakozási pontot tesz lehetővé. Helytakarékos és mivel a csatlakozások a chip alatt helyezkednek el, rövidebbek a vezetési utak és a hőelvezetési tulajdonsága is jobb. Hátránya azonban, hogy a tokozás javítása, újra forrasztása nehézkes éppen úgy, mint a forrasztási pontok megfelelőségének ellenőrzése. Az ellenőrzés egyik eszköze a síkcsiszolat-készítés. [11] [12]



1. ábra BGA Felépítése

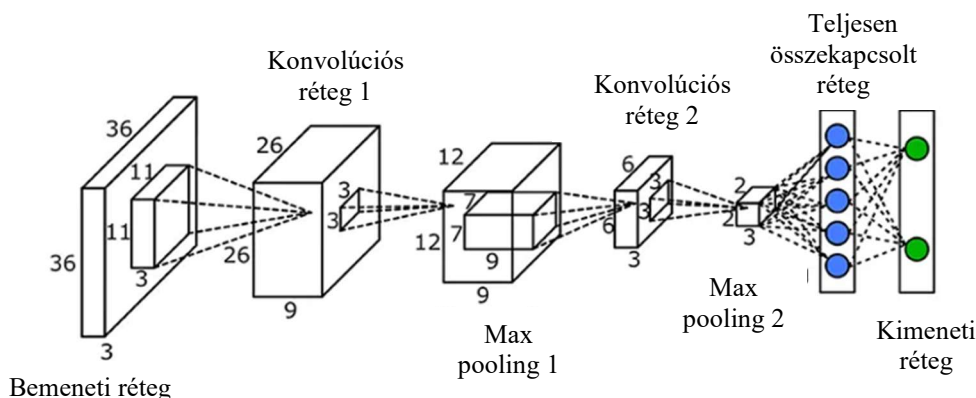
2.4 A mesterséges intelligencia

Az ember évezredek óta kutatja és próbálja megérteni, hogyan működik saját gondolkodása. Hogyan zajlik a körülöttünk lévő világ felfogása és megértése, valamint hogyan történik a válaszreakciók megalkotása, a környezetünk manipulálása, az események megjósolása.

A mesterséges intelligencia – mint tudományterület - az emberi gondolkodás megértése mellett a hasonlóan intelligens entitások létrehozásával foglalkozik. Számítalan leírás és definíció áll rendelkezésre, amelyek a mesterséges intelligenciát próbálják pontosan meghatározni. Az egyik ilyen megközelítés az, amelyben a gépet egy vitathatatlanul intelligens entitáshoz hasonlítjuk, ez a Turing-teszt. [13] [14] [15]

2.5 Objektum felismerés konvolúciós neurális hálóval (CNN)

A konvolúciós neurális hálók több rétegből állnak, ezek a bemeneti réteg, a konvolúciós réteg, a max pooling réteg, a fully-connected réteg és a kimeneti réteg.



2. ábra Konvolúciós neurális háló

Egy kép egyszerűen egy egydimenziós vektorra alakítható, ha végig vezetjük egy összetett neurális hálón, de nagy méretű, több szint tartalmazó kép esetén ez komoly számítási kapacitást igényel. Egy normál kép RGB színeket tartalmaz, ahol a színskála 0-255-ig terjed. Egy kép mérete általánosan: $n \times n \times 3$, ahol „n” a pixelek száma.

A CNN a konvolúciós függvényt használja a kép méretének csökkentésére adott kernel szűrő skaláris szorzatával. A konvolúció általános kifejezése:

$$g_{x,y} = \omega \cdot f_{x,y} = \sum_i \sum_j \omega_{i,j} f_{x-i,y-j} ; \quad (1)$$

ahol $g_{x,y}$ a szűrt kép, $f_{x,y}$ az eredeti kép, ω pedig a kernel szűrő.

Ennek következtében a kép mérete csökken, de a kép tulajdonságai megmaradnak, így a kép feldolgozása lényegesen könnyebbé válik.

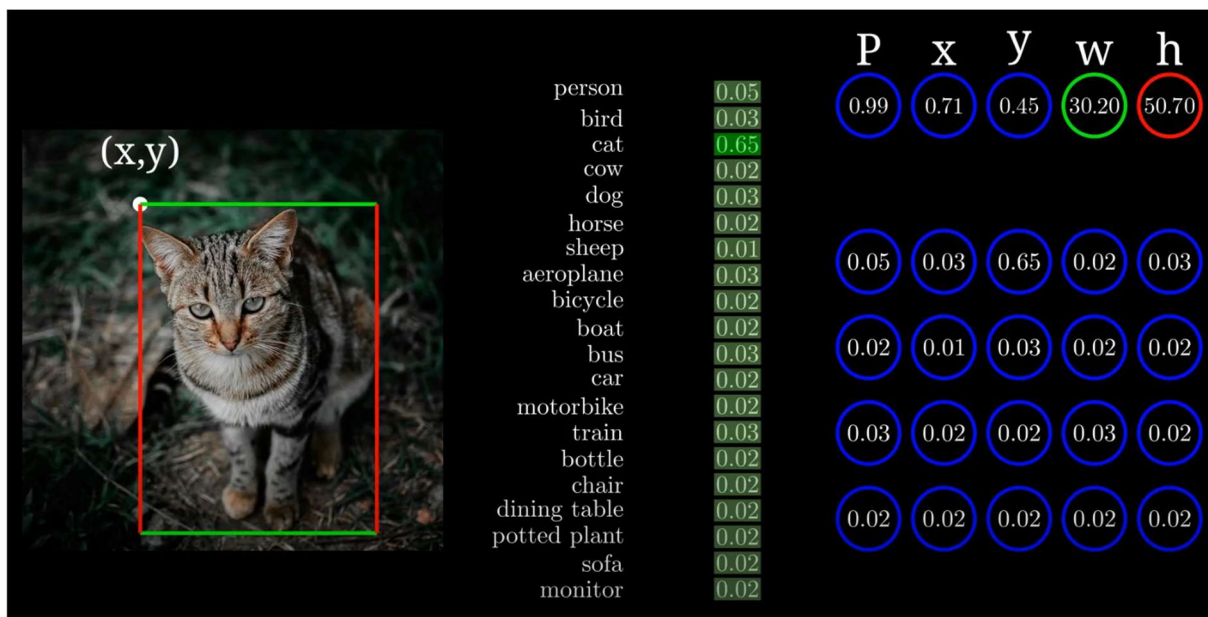
A konvolúciós réteg tehát egyszerűsítette a bemenetre adott képet úgy, hogy annak tulajdonságai megmaradtak. A következő réteg a kép térbeli méretét csökkenti, ez a pooling réteg. Ennek előnye, hogy a kép méretének csökkenésével csökken a feldolgozásához szükséges számítási teljesítmény. Ennek a folyamatnak kétféle eljárási módja van, a max pooling és az average pooling.

Max pooling esetén az adott mező legnagyobb értékét, míg az average pooling-nál a mezőben szereplő számok számtani közepét használja a további számításhoz. Jellemzően a max pooling jobban teljesít, mint az average pooling.

Az eddig történeteket Feature Extraction-nak nevezik. A folyamat eredménye egy oszlopvektorba sűrítendő, ami átadható a neurális hálónak. A folyamat második felét osztályozásnak nevezik. [16] [17] [18] [19]

2.5.1 A YOLO modell (You Only Look Once)

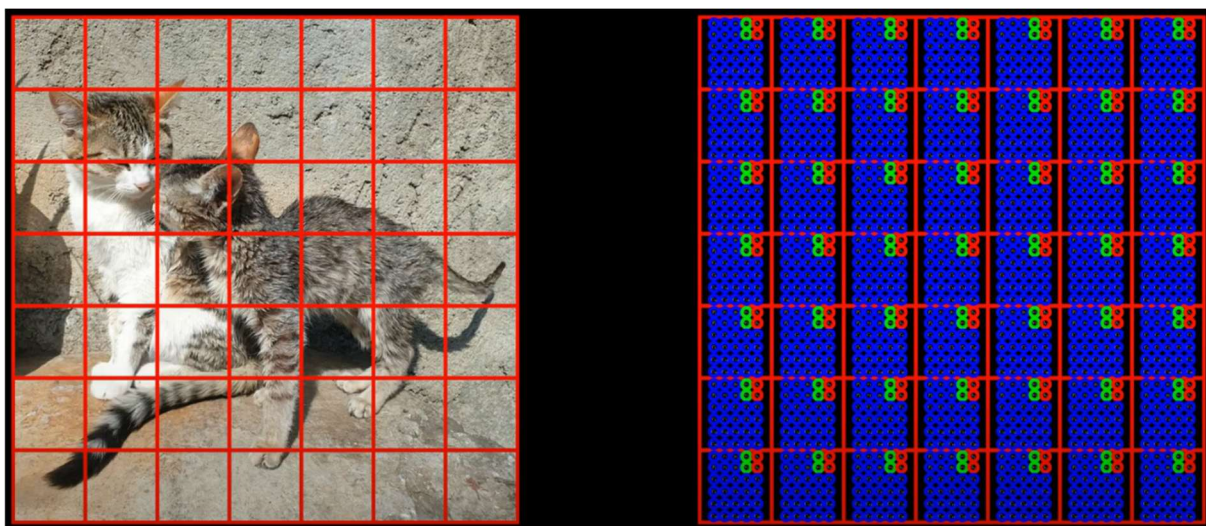
Objektumdetektálás esetén két fő feladat van. Meg kell neveznünk a képen szereplő objektumokat és meg kell határozni azok elhelyezkedését. Egy objektum esetén elegendő egy neuron annak meghatározására, hogy a képen látható az objektum vagy sem. Példaképpen, ha a neuron értéke nagyobb, mint 0.5, az objektum szerepel a képen, amennyiben 0.5-nél kisebb az érték, az objektum nem szerepel a képen. Amennyiben az objektum szerepel a képen, szükségünk van annak bal felső sarkának koordinátaira. Ezt követően szükséges a határoló négyzet szélessége és magassága. Amennyiben ezek az adatokat ismertek, a határoló négyzet könnyedén megrajzolható. Az objektum sarkának koordinátái (x,y) további kettő, valamint a határoló négyzet oldalainak hossza és magassága szintén további kettő neuront igényel. Az objektum felcímkézése szintén szükséges, ezért további 20 neuront adunk hozzá, amelyek 20 különböző objektum osztályozására szolgálnak. [20] [21]



3. ábra YOLO modell neuronjai

A „P” érték mutatja, hogy van-e objektum a képen, „x” és „y” az objektum középpontját, „w” és „h” pedig a határoló négyzet éleinek hosszát. A további 20 érték a detektált objektum osztályozására szolgál.

Ha két objektumot szeretnénk detektálni, az első sort megduplázva további 5 neuront illesztünk be. Így tehát 30 neuronra lesz szükség. Ha kettőnél több objektum szerepel a képen, amelyek eltérő osztályba tartoznak, mást technikát kell alkalmazni. Az egész képet egy 7x7-es ráccsal osztjuk fel. Minden rács két objektum detektálásáért felelős. Ezt követően 7x7x2, vagyis 98 objektum detektálható. A művelethez $7 \times 7 \times (2 \times 5 + 20) = 1470$ neuron szükséges, ahol 7x7 a teljes rács és 30 érték a két határoló négyszög minden egyes rácstra. Amennyiben egy objektum több rácspan is szerepel, az objektum középpontját tartalmazó rács lesz felelős az objektum érzékeléséért. [22] [23]



4. ábra A YOLO modell rács-felosztása

2.5.2 YOLO Loss Function – Veszteségfüggvény

A YOLO modell veszteségfüggvénye a következő összefüggéssel írható le:

$$\begin{aligned}
 \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \\
 + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right] \\
 + \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2 \\
 + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2 \\
 + \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
 \end{aligned} \tag{2}$$

Ahol:

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \tag{3}$$

Az első összegzés a rácsok összegét adja, a második a rácsokban található határoló négyszögeket, x és y jelöli a középpontok koordinátáit, a valós és a jósolt értékek négyzetes hibák összegeként szerepelnek. Mivel nem minden rács tartalmazza a detektálni kívánt objektumot szükséges egy kapcsolóként működő elem. A hiba számítása előtt lévő 1-es érték 1, ha van objektum, 0, ha nincs. Amennyiben nincs objektum, az egész kifejezés értéke 0 az adott rácsra vonatkozóan.

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right] \quad (4)$$

A kifejezés második tagja hasonló az előzőhöz. Itt kerül meghatározásra a határoló négyszög szélességének és magasságának hibája. A kifejezésben a hiba számítása az értékek négyzetgyökével történik. ennek oka, hogy minél kisebb a határoló négyszög, annál nagyobb pontatlanságot eredményezne, ha az első sor szerint járnánk el.

$$\sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2 \quad (5)$$

$$\lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2 \quad (6)$$

Itt kerül számításra a modell konfidenciája, vagyis az, hogy mennyire biztos abban, hogy egy objektum egy adott osztályhoz tartozik. Az 1-es szorzók egymás ellentettjei és azt hivatottak jelezni, hogy a rácsban szerepel-e objektum. Amennyiben nem, az első kifejezésben 0, a másodikban 1 lesz. Ekkor figyelembe vesszünk egy λ_{noobj} szorzót, amelynek értéke 0.5. Ezzel csökkentjük a hiba jelentőséget abban az esetben, ha a rács nem tartalmaz objektumot.

$$\sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \quad (7)$$

Az utolsó sor a klasszifikációs veszteség. Ez „bünteti” a modellt, ha a prediktált osztály nem felel meg az elvártnak.

Az első két sorban λ_{coord} értéke 5. Ennek funkciója növelni ezeknek a hibáknak az értékét és kényszeríti a modellt, hogy jobban fókuszáljon a határoló négyszögek helyes jóslására. [24]

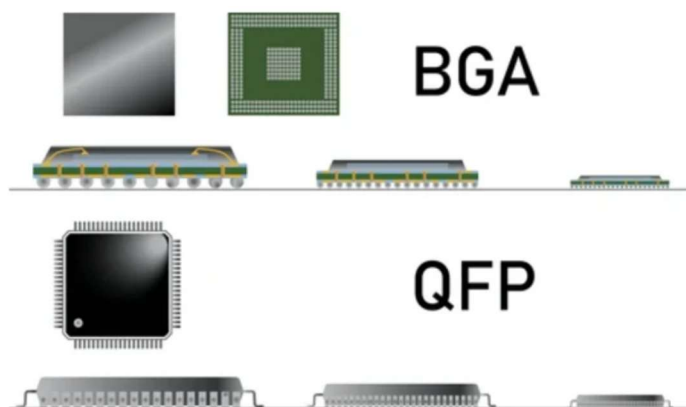
3. Anyag és módszer

Ebben a fejezetben bemutatásra kerül a feldolgozni kívánt BGA keresztcsiszolatokról készült képek előállításának menete, a feldolgozást végző mesterséges intelligencia tanításának gyakorlati folyamata és a kiértékelés logikája, valamint a program használatát lehetővé tevő grafikus felhasználói felület programozásának menete.

3.1 Csiszolatkészítés jelentősége

A dolgozat témájának alapjául szolgáló BGA golyós rácsömb típusú integrált áramköri tokozás forrasztásainak vizuális ellenőrzése fontos feladat. Alkalmazásának előnye a QFP (Quad Flat Package – lapos csomag) tokozással szemben, hogy kevesebb helyet foglal a nyomtatott áramköri lapon. Hátránya viszont, hogy a forrasztási hiba forrasztott állapotban nehezen ellenőrizhető, nem látható azonnal. Ha nyomtatott áramköri lap valamilyen mechanikai hatás éri és a lap meghajlik, a forrasztás könnyen sérülhet, repedhet. Ennek okán a BGA típusú tokozások esetében a sarkokban elhelyezkedő lábakat nem érdemes a programozás során használni, azokhoz funkciót társítani. Ezek a mechanikai hatások a termék élettartama során bármikor bekövetkezhetnek, számomra és a dolgozat szempontjából a gyártás közben keletkező esetleges problémák feltárása releváns. Emellett a forrasztás során hibát okozhat az alapanyag minősége vagy a technológiai paraméterek nem megfelelő megválasztása, például a rosszul beállított forrasztási hőprofil. [25] [26] [3]

A két fő hibaszempont tehát - ami igazolja a széria vizsgálat szükségességét - a repedések és a forrasztáson belüli üres térrészek (void) feltárása.



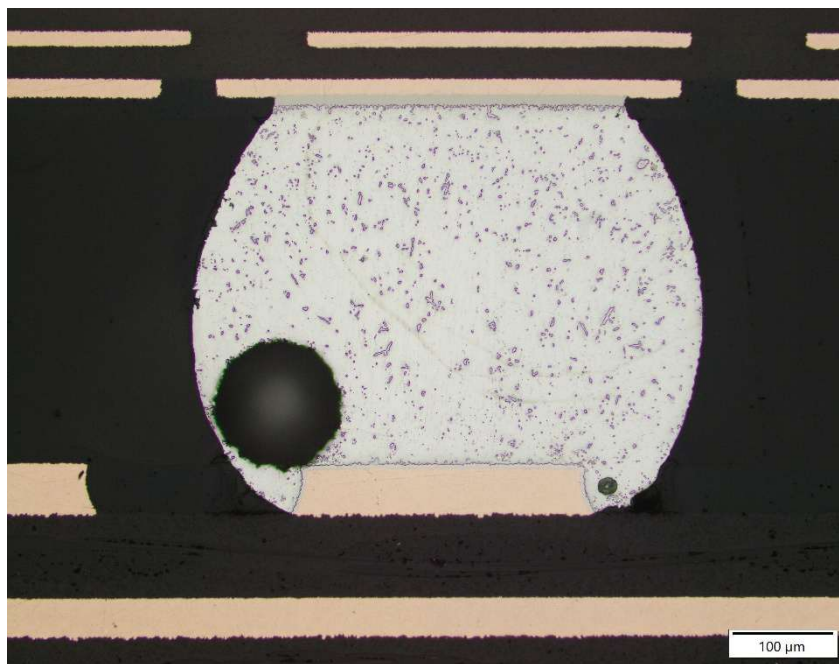
5. ábra A BGA és QFP tokozás

3.2 Csiszolatkészítés menete

A mesterséges intelligencia modell tanító adataként szolgáló képek jelentős részét én készítettem, a következőkben olvasható a képek alapjául szolgáló csiszolatok elkészítésének folyamata.

A csiszolatok készítése során fontos, hogy a vizsgálni kívánt alkatrész a folyamat során ne szenvedjen semmilyen sérülést vagy deformációt, hiszen ez meghamisítaná a vizsgálat eredményét. Ennek elkerülésére a nyomtatott áramköri lapból való kivágást megelőzően epoxi gyantával rögzítettem a BGA-t a nyomtatott áramköri laphoz. A gyanta kötési idejének letelte után egy kézi csiszoló és gravírozó multifunkciós gépet használtam gyémánt koronggal a BGA kivágásához. A gyémánt korong kemény anyagok vágására szolgál, mint a beton, kő, porcelán, kerámia és kemény epoxi. A kivágott mintát a vizsgálni kívánt síkkal párhuzamosra csiszolva mintatartóba helyeztem, majd ismét felöntöttem epoxi gyantával. A felöntési folyamat során kétkomponenses epoxi resint használtam, amely kimondottan metallográfiai vizsgálatokhoz javasolt. A térhálósodási folyamat közben hő keletkezik, időként gondoskodnom kellett a mintatartók megfelelő hűtéséről. 10-12 óra kötési idő után a mintát eltávolítottam a mintatartóból és elkezdtem a vizsgálni kívánt sík csiszolással való megközelítését. A csiszolást egytárcsás vizes csiszológépen végeztem, a síkot közelítve egyre finomabb szemcséjű csiszolópapírt helyeztem a tárcsára. A folyamat során a csiszolás előrehaladását mikroszkóp alatt ellenőriztem, ezzel megelőzve az esetleges túlhaladást. A vizsgálati síkot elérve a mintát políroztam, hogy a csiszolás során keletkezett karcok eltűnjenek a minta felületéről. A polírozást géppel végeztem 9 μ szemcsemérettől indulva a 3 μ -on át a kevesebb mint 1 μ -ig. A polírozás során mindhárom lépcsőnél szuszpenziós folyadékot is használtam a jobb eredmény érdekében. [27] Ezt követően a mintát lemostam és szárítottam, készen állt a képi dokumentálásra. A képek készítéséhez Olympus GX71 típusú inverz mikroszkópot használtam, ezzel elkerülve a minta-objektív párhuzamosságának problémáját. A képeket 200-szoros nagyításban készítettem, a fájlokat jól strukturáltan mappákba rendeztem, a képek nevét a BGA kiosztása szerint adtam meg. A mappastruktúrának és a kép nevének később jelentősége lesz a program használatánál.

A teljeskörű vizsgálathoz egy BGA-n 6 síkot csiszoltam, tehát 6 lábsort tettem láthatóvá. Nagyobb alkatrészek esetén egy sor 20-25 csatlakozási pontot jelent, ebből következik, hogy egy alkatrész esetén legalább 120 képet kell ellenőrizni. Ez indokolja az ellenőrzési folyamat automatizálását.



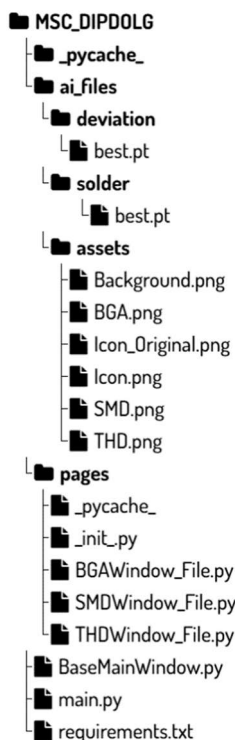
6. ábra BGA láb keresztcsiszolati képe

3.3 Program

A program fejlesztéséhez a Visual Studio Code ingyenes nyílt forráskódú kódszerkesztőt használtam és Python programozási nyelvet. A programhoz használt fájlok átláthatósága és könnyebb kezelése érdekében a következő struktúrát vezettem be:

Létrehoztam egy mappát, amely az egész projektet foglalja magába, ezen belül további almappákat a mesterséges intelligencia modell fájljaihoz, a projektben használt mellékletekhez (képek, ikonok) és a programban megjelenő ablakokhoz. A main.py fájlból futtatható a program. A BaseMainWindow.py a főablak, amelynek a tartalma cserélődik ablakváltáskor, a requirements.txt pedig azokat a könyvtárakat tartalmazza, amelyek telepítése szükséges a program használatához. A requirements.txt segítségével a python csomagkezelő egyetlen paranccsal képes telepíteni a szükséges könyvtárakat. A következő ábra a programhoz használt mappastruktúrát mutatja. [28]

A program fejlesztésének folyamatát a bemutatás során két részre bontom. Az első felében ismertetem a grafikus felhasználói felület mögött álló forráskód működését és felépítésének logikáját. A második felében a mesterséges intelligencia modell tanításának folyamatát és annak implementálását mutatom be. Ez a sorrend a fejlesztés kronológiájával megegyezik.



7. ábra A projekt struktúrája

3.3.1 Grafikus felhasználói felület (GUI)

A program fejlesztését a grafikus felhasználói felület tervezésével kezdtem. Ennek a megközelítésnek a célja az volt, hogy legyen egy olyan felület gombokkal ellátva, amelyekhez a későbbiekben funkciókat rendelhetek. [29] [30]

3.3.2 A main() függvény

A GUI elkészítéséhez a PyQt5 könyvtárat használtam. A PyQt5 egy widget-eszköztár, amely alkalmazásfejlesztési keretrendszerként szolgál. Grafikus felhasználói felületek esetén jellemzően egy ablakban fut a program. Az ablak létrehozásához szükséges a QApplication, ami az eseményeket kezeli. Szemléletes analógiával élve, a QApplication a színház épülete. Ezt követően létrehoztam a színpadot, vagyis a BaseMainWindow-t. Ez a főablak, amely tartalma oldalváltáskor cserélődik, hordozza a program nevét és ikonját, a teljes program háttereként szolgáló képet és itt definiáltam a megjelenő ablak méretét is. Elsőként a WelcomePage jelenik meg, mint első színházi jelenet.

Ezt a felépítést tartalmazza tehát a main() függvény. Belépési pontként szolgál, összerakja és elindítja az eseményciklust.

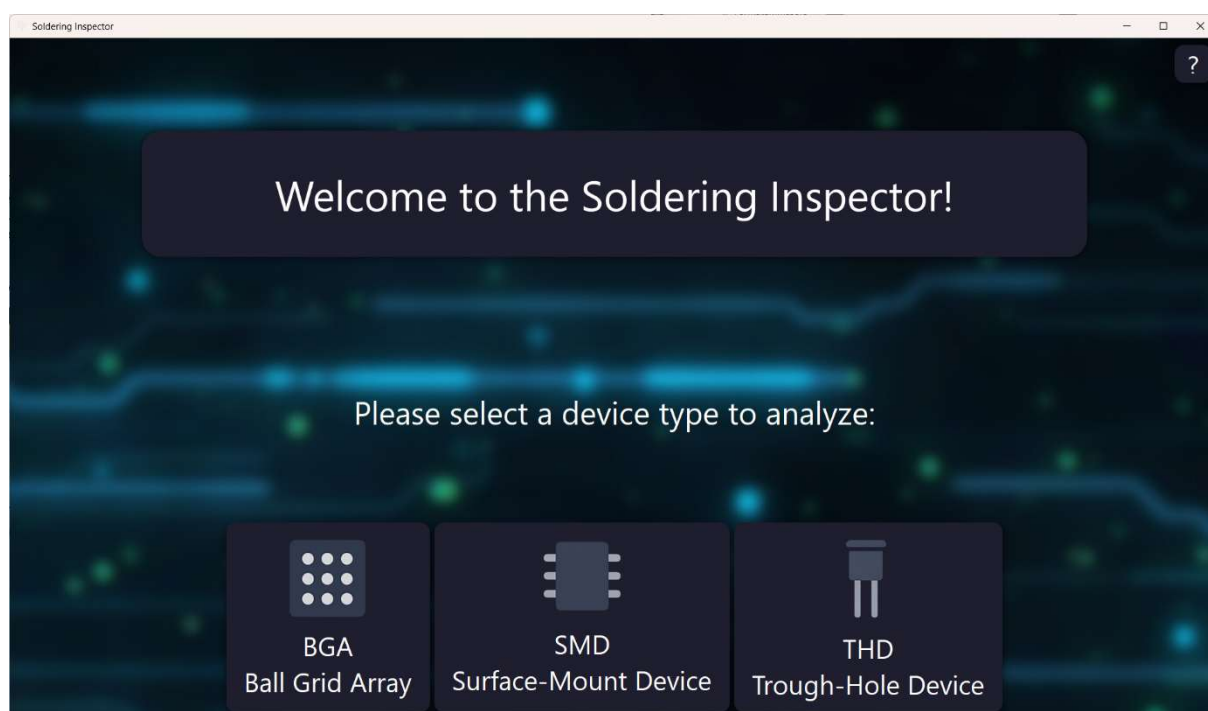
3.3.3 Az üdvözlő ablak (WelcomePage)

Ez az osztály az alkalmazás nyitóképernyőjét foglalja magába. A megjelenéshez sötétebb, modern, árnyékolt dizájnt választottam. Az elemek elrendezéséért a QVBoxLayout és QHBoxLayout osztályok felelnek, egymás alá/mellé rendezik az elemeket. A jobb felső sarokban egy súgó gomb található. Ennek leírása, hogy a program egy forrasztás vizsgáló alkalmazás. Középen olvasható az üdvözlőszöveg, alatta pedig az utasítás: Kérlek válassz egy eszköztípust a vizsgálathoz. Itt 3 alkatrésztípust választhat a felhasználó:

- BGA Golyós rácsömb
- SMD felületszerelt alkatrész
- THD furatszerelt alkatrész

A választógombokat nagy méretűre állítottam, hogy a feliratuk és a rájuk helyezett ikonok is jól láthatóak legyenek. A gombokon elhelyezett ikonokat mesterséges intelligenciával készítettem. Keresztcsiszolati képeket mutatva kértem, hogy generáljon belőlük ikonokat. Az oldalon való navigációt segíti, hogy a gombok megjelenéséhez lebegő effektet is rendeltem, a kurzort a gombok fölé húzva színük megváltozik.

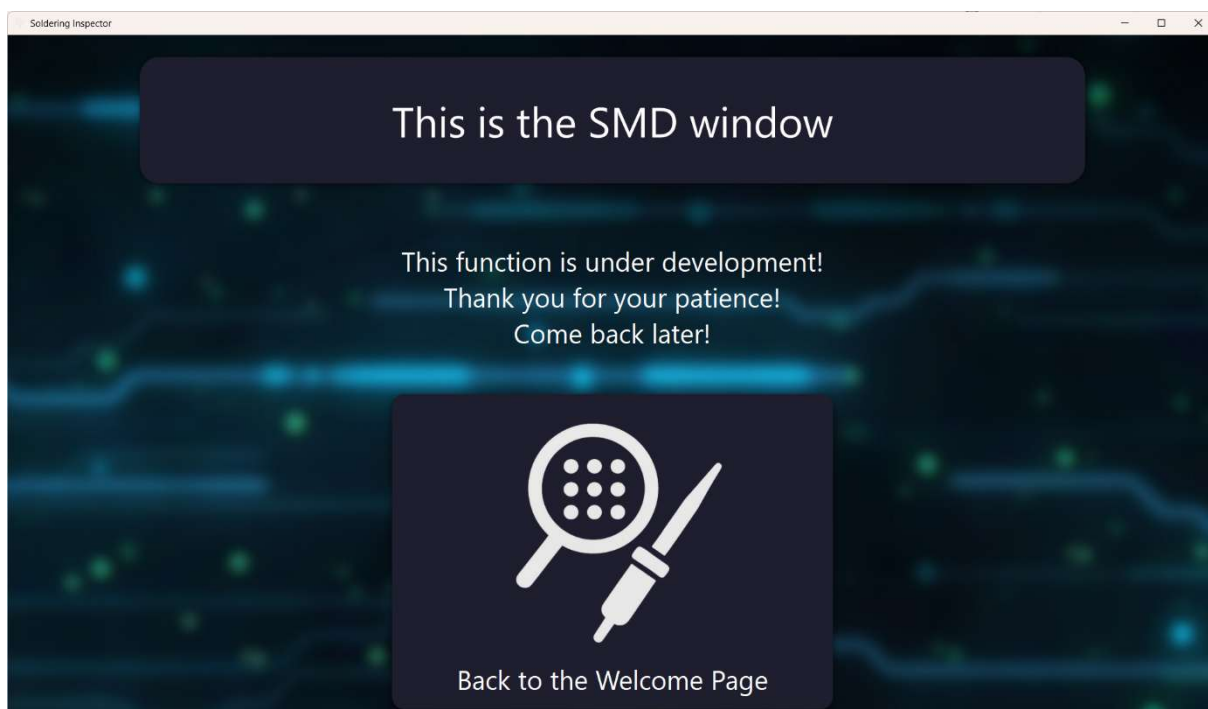
A WelcomePage osztályban az ablak felépítése és a dizájn megalkotása mellett, itt kezelem a gombokra való kattintás eseményeit is. Valamely gombra kattintást követően a felhasználót átirányítom a megfelelő oldalra. Az alkalmazás jelenlegi állapotában csak a BGA forrasztások vizsgálata érhető el, az SMD és THD ablakok fejlesztés alatt állnak. Erre vonatkozóan az SMD és THD ablakok kiválasztását követően ezt a felhasználó tudtára is adom.



8. ábra Üdvözlőablak

3.3.4 SMD és THD ablakok

Az SMD és THD ablakok tehát fejlesztés alatt állnak a dolgozat készítésének idejében. A helytartó egy tájékoztató szöveg, miszerint: Ez a funkció fejlesztés alatt áll! Köszönöm a türelmet! Gyere vissza később! Az oldal megjelenése megegyezik az eddigiekben olvasottakkal, egy nagy méretű gomb teszi lehetővé az üdvözlőoldalra való visszatérést. A gombon erre vonatkozó felirat és az alkalmazás ikonja látható.



9. ábra Az SMD ablak

3.3.5 A BGA ablak

Ebben az alfejezetben bemutatom a program BGA-elemző moduljának felhasználói felületét.

A BGAWindow osztályban inicializálom a felhasználói felületet. Az ablakot 3 részre osztottam, bal, közép és jobb oldali panelekre. A bal oldali panelen helyeztem el a funkciógombokat QVBoxLayout osztállyal egymás alá, melyek rendre:

Load Image: Ez a gomb felel egy darab kép megnyitásáért.

Load Image Folder: Lehetővé teszem képek egy mappájának való megnyitását.

Run Analysis: Egy darab kép kiértékelésének indítása.

Auto-Process: Automatikus képfeldolgozás egy mappában található képekre.

Save Result: Az eredmény mentése

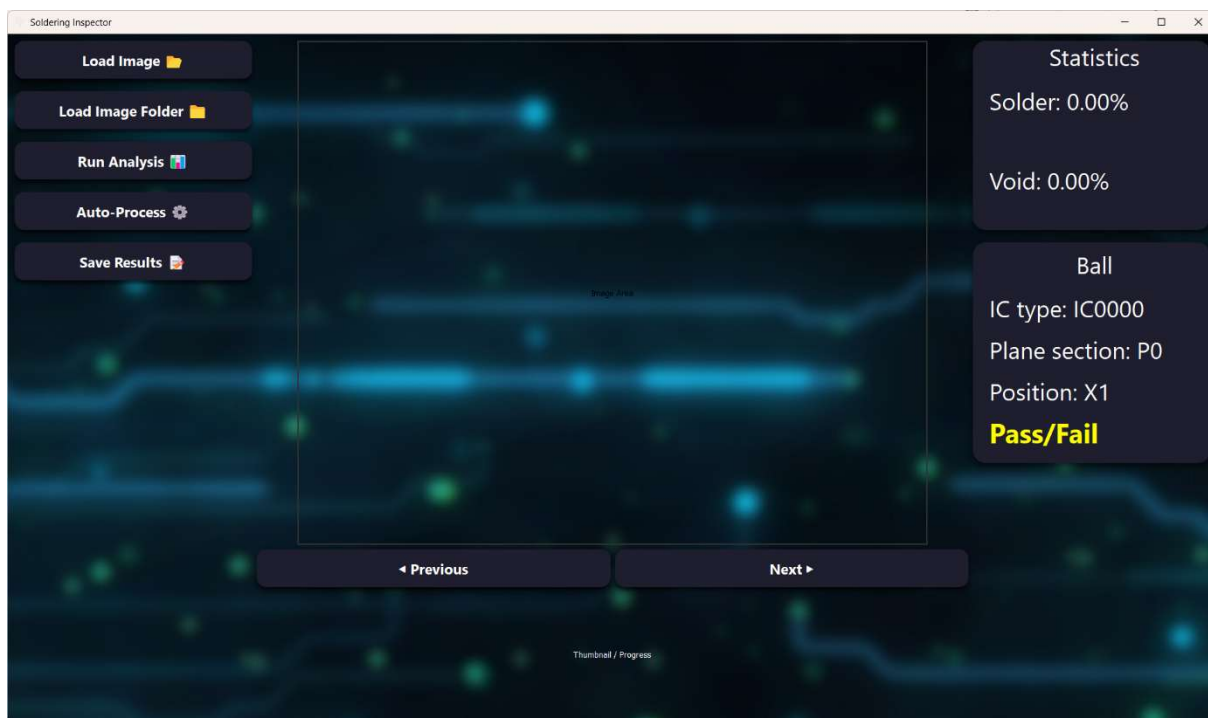
A funkciók működését a következő fejezetben mutatom be.

A gombok megjelenése egységes a korábban ismertettekkel.

A középső panel szintén egy QVBoxLayout-ot tartalmaz. Itt definiáltam a képek megnyitására szolgáló felületet, alatta a navigációs gombokat, amelyek a mappa megnyitását követően lehetővé teszik a képek közötti lapozást. Alul egy további címkét helyeztem el helytartóként, amely a program továbbfejlesztésének szempontjából megkönnyíti a bővítmény elhelyezését. A következő fejezetben lesz erre példa.

Végül létrehoztam a jobb oldali panelt, amely információt ad az aktuálisan megnyitott képről és eredmények azonnali láthatóságát teszi lehetővé. Az előzőekhez hasonlóan a QVBoxLayout osztályt használtam. Felülre került a Statistics nevű szövegdoboz, amely a forrasztások százalékos arányát mutatja. Alatta egy másik szövegdobozban a vizsgált integrált áramkör típusa, a csiszolati sík és a vizsgált láb neve jelenik meg továbbá az analízis futtatását követően megjelenik egy indikátor felirat, a zöld színű Pass vagy a piros színű Fail. A program indításánál ez sárga Pass/Fail feliratként jelenik meg.

A három panel egymás mellé illesztése a QHBoxLayout osztállyal végeztem el, amely mint fő kiosztás szerepel a kódban. [31]



10. ábra A BGA ablak

3.4 A BGA modul működése

Ebben az alfejezetben bemutatom a program BGA-elemző moduljának működését.

A program írása során az objektumorientált programozás irányelveit igyekeztem követni, azonban felfedezhetőek benne strukturált programépítési alapelveknek megfelelő részek is. A BGAWindow ablak inicializálását követően tehát rendelkezésre állnak a gombok, amik indítják az eseményeket, és a felületek, ahol az adatok megjelenhetnek.

Következő lépésként funkciókat rendeltem a gombokhoz, eseményeket csatoltam a gombok megnyomásához, ehhez függvényeket hoztam létre.

3.4.1 Load Image

A `load_image()` függvényben a `QFileDialog` osztály `getOpenFileName` függvényét használtam egy kép megnyitásához. A kép megjelenési helye a korábban említett középső panelrész, ahova a kép méretarányának megtartásával kerül be. A kép megnyitását követően a jobb oldali panelen az IC típusa, a csiszolat síkja, és a láb megnevezése megjelenik. Az ehhez tartozó információkat a jól strukturált mapparendszerből nyерem ki, itt van jelentősége annak, amit a korábbi Csiszolatkészítés fejezetben írtam. A BGA csiszolatokról készült kép

megnevezése a golyó pozícióját és a nagyítás mértékét is hordozzák a következő formában: A1_x200. Mivel ezek a képek minden esetben 200-szoros nagyításban készülnek, számomra a fontos információ az első tag. A fájl nevéből először levágtam a fájl kiterjesztését, ami .png, .jpg vagy .bmp. Ezt egy új változóban tárolva tovább vágtam az „_” szimbólumnál, így megkaptam tisztán a golyó nevét. Egyedi kép esetén az IC típusa és a csiszolati sík ismeretlen, feltételezve, hogy egyedi képnél a felhasználó ismeri a vizsgált mintát. Egy kép esetén az előző és következő gombok inaktívak.

3.4.2 Load Image Folder

A `load_image_folder()` függvény a `getExistingDirectory` függvénnyel nyitja meg a képeket tartalmazó mappát. A mappa tartalmából csak a képformátum kiterjesztéssel rendelkező elemek kerülnek megjelenítésre, nem okoz problémát, ha a mappában más fájltypus is jelen van. A mappa megnyitását követően a jobboldali panelen megjelennek a golyóhoz tartozó információk. Ehhez a `show_image()` segédfüggvényt hoztam létre. A golyó neve az előzőekben ismertetett módon áll elő, a csiszolati sík megnevezése és az IC típusa pedig a fájl elérési útvonalából, ezért fontos, hogy a mappák felépítése egységes legyen és a következő mintát kövesse:



11. ábra BGA képek szükséges mappastruktúrája

A megnyitandó mappák a csiszolati síkkal ellátott mappák: P1 vagy P2, ahol P (Plane) a síkra utal.

A mappa megnyitását követően a navigációs „előző” és „következő” gombok aktívak, használhatók a képek léptetésére.

3.4.3 „Előző” és „következő” navigációs gombok

A navigációs gombok használatához a mappa betöltését követően létrehoztam egy `current_image_index` nevű változót, amely a jelenlegi kép indexelését jelenti. Ha van betöltve mappa, akkor a „következő” gomb megnyomását követően a `current_image_index` változó értéke eggyel nő majd maradékosan el is osztom a mappában található összes elem értékével. Ennek a logikának köszönhetően az utolsó kép után a sor előről indul és ismét az első elem jelenik meg. Az előző gomb megnyomásánál a különbség mindössze annyi, hogy a `current_image_index` értékét nem növelem, hanem eggyel csökkentem.

3.4.4 Run Analysis

A `run_analysis()` függvény az alkalmazás motorja. Felépítése a következő:

1. Előkészítés

A függvénybe való belépésnél ellenőrzöm, hogy a felhasználó töltött-e be képet. Amennyiben nem, a következő üzenet jelenik meg a terminálon: No image loaded to analyze. Ha van kép akkor azt betöltöm.

2. Képfeldolgozás

A feldolgozásért két előre feltanított mesterséges intelligencia modell felel. Az első modell a forrasztási hibákat keresi, a második a forrasztási felületet vizsgálja. Mindkét modell esetében az eredmény egy annotált (címkézett, feljelölt) kép. A modellek tanításáról a későbbi fejezetben írok.

3. Eredmények összegzése

A két képet a megjelenítéshez 50-50% összefésülöm, így szemmel látható, hogy a mesterséges intelligencia mely területet jelölte ki a forrasztásnak és hol fedezett fel forrasztási hibákat, repedést és void-ot.

4. Százalékos eredmények

A második modell alapján összegzem a pixeleket, amelyek a forrasztási felületen belülre esnek, majd hasonlóan összegzem a forrasztási hibákat lefedő pixeleket. A két értéket összevetve meghatározom, hogy a forrasztás teljes területéhez képest hány százalék a void és a repedések arány. A voidok és repedések közül csak azok kerülnek bele a számításba, amelyek átfedésben vannak a forrasztással. Erre azért van szükség, hogy a forrasztáson kívülre eső

esetleges rossz predikciók semmiképpen ne kerüljenek bele az eredménybe. A logikát ebben az esetben egy „és” logikai művelettel oldottam meg, a void-ok és repedések közül csak azok számítanak, amelyek átfedésben vannak a forrasztanyaggal. A számítást követően az eredményt megjelenítem a terminálon és a grafikus felhasználói felületen is a korábban bemutatott Statistics szövegdobozban, az annotált képet pedig a képnek fenntartott területen adom vissza a felhasználónak. Az eredmény számítását követően megjelenik a Ball szövegdobozban az indikátor felirat is, amely előre definiált szabály alapján dönt a forrasztás milyenségéről (Fail / Pass).

3.4.5 Auto-Process

A `run_batch_analysis()` függvényt a kötegetben való feldolgozáshoz hoztam létre. Lényegi funkcióját tekintve ugyan az mint az előző, azonban automatikusan több példányon végzi el a vizsgálatot. Nézzük lépésről lépésre:

1. Előkészítés

Először ellenőrzöm, hogy a felhasználó töltött-e be mappát. Ha nem hibát jelezek a terminálon, ha igen inaktívvá teszem a kezelőgombokat, hogy a feldolgozás alatt a felhasználó ne tudjon hibát okozni. A folyamat során a már korábban említett helytartóként funkcionáló címkén megjelenítek egy folyamatjelző sávot, egy progress bar-t. Ezzel adok visszajelzést a felhasználónak, hogy hányadik képnél tart a feldolgozás, a progress bar osztása a mappában szereplő képek számától függ, rajta egy százalékos érték is tájékoztatja a felhasználót a folyamat állapotáról.

2. Kimeneti mappa

Automatikusan létrehozok egy kimeneti mappát „results” néven a képek mellett, amelybe a feldolgozott annotált képek kerülnek majd és egy „json” fájl a képekhez tartozó adatokkal.

3. Képek feldolgozása

A képek feldolgozásánál egyesével iterálok minden képen. Minden kép esetén frissítem a Statistics panelt az IC típusát, csiszolati síkját és nevét illetően. Meghívom a `run_analysis()` függvényt, amely az előzőekben ismertetett módon kiszámítja a százalékos értékeket és eldönti, hogy a forrasztás megfelelt-e vagy sem. Ezt követően a „progress bar” eggyel tovább halad. Az

annotált képek a results mappába kerülnek, a hozzájuk tartozó adatok, mint az IC típus, sík, név, void/solder % arány, megfelelés pedig egy json fájlba a képek elérési útjával összekötve.

4. Befejezés

A „progress bar” eltűnik, a gombok újra aktívvá válnak, a terminálon pedig megjelenik egy üzenet a feldolgozás és a mentés befejezéséről.

3.4.6 Save Results

Az eredmények előálltak, azonban szerettem volna egy átláthatóbb, felhasználóbarát megoldást azok prezentálására, ezenfelül még nem foglalkoztam a program eredményeinek felhasználását illető felelősségvállalással. Ezen indokok vezettek rá arra, hogy létrehozak egy függvényt, amely PDF-ként exportálja és összegzi az eredményeket azután, hogy a vizsgálatot végző személy (operátor) egy felugró ablakban megadja a nevét, ezzel hitelesítve az eredményeket és vállalva a felelősséget. Az általam készült alkalmazás nem épül szabványokra. Ez egy segédprogram az analízist végző személy számára a gördülékenyebb munkavégzés érdekében, a felelősség minden esetben a felhasználót terheli.

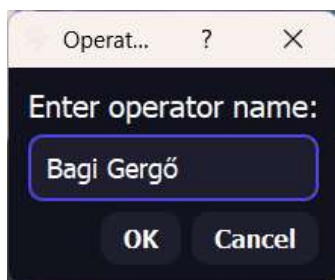
Az `on_save_results_clicked()` függvény felépítése a következő:

1. Előkészítés

Ellenőrzöm, hogy valóban megtörtént-e az analízis, vagyis, hogy van-e results mappa és hogy benne vannak-e a képek és a json fájl. Ha nincs, hibával leáll, ha van az operátor azonosítása következik.

2. Operátor azonosítása

Egy felugró ablakban bekérem a felhasználó nevét. Ez biztosítja, hogy az elkészült jelentésben szerepeljen a vizsgálatot végzett személy neve. A felugró ablakban a név beírása kötelező, másként a mentés nem indul el.



12. ábra Operátor ablak

3. Mentés

A felhasználó kiválaszthatja a mentés helyét, a riportot elnevezheti.

4. PDF generálás

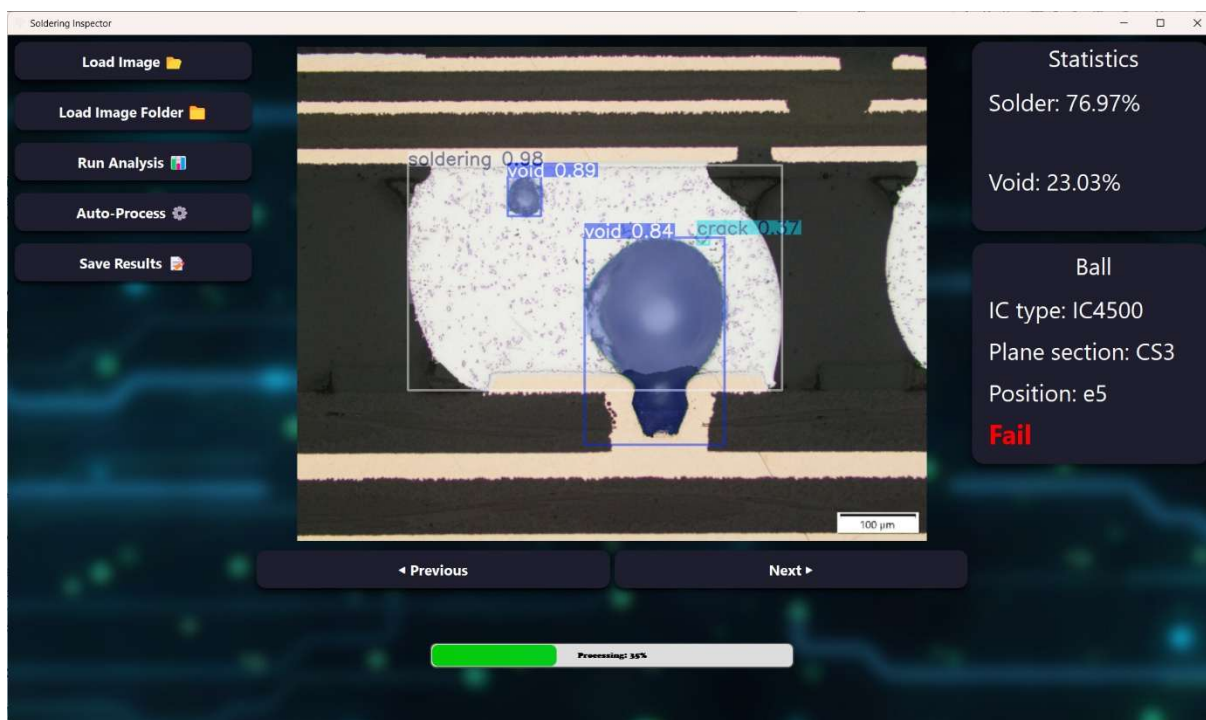
A PDF generálásához egy másik függvényben valósítottam meg, amit itt hívok meg

A függvény neve `export_pdf_from_json()`.

Ez a függvény először beolvassa a korábban mentett json fájlt. Az inicializálásnál definiálom az A4-es oldalméretet. Egy „story” nevű lista változóban összegyűjtöm az össze adatot, amit szeretnék a PDF-ben megjeleníteni. A borítón az alkalmazás neve jelenik meg az alkalmazás ikonjával és a vizsgálatot végző operátor neve. A következő oldalakon táblázatos formába szedve jelennek meg az egyes forrasztásokhoz tartozó adatok, a táblázat alatt pedig az annotált kép. Minden eredmény külön oldalon szerepel. A dokumentumhoz oldalszámozást is rendeltem.

5. Visszajelzés

A mentés sikerességéről a terminálon érkezik információ.



13. ábra BGA ablak programfutás közben

Ezzel a programműködés bemutatásának végéhez értem. Számos fejlesztési irány vehető számításba az alkalmazás fejlesztését illetően. Ilyen az SMD és THD alkatrészek ablakának fejlesztése, új funkciók bevezetése, mint például a kézi annotálás, de nem utolsó szempont a PDF generálás kezdetleges megjelenésének továbbfejlesztése sem.

3.5 A mesterséges intelligencia tanítása

A vizsgálathoz az előzőekben ismertetett YOLO (You Only Look Once) előre tanított mesterséges intelligencia modellt használtam. A sajátcélú felhasználáshoz újra tanítottam a modellt. Ehhez szükségem volt egy saját tanító adatbázisra. Az eredmény minősége egyenesen arányos a tanításhoz használt elemek számával, ennek okán összegyűjtöttem 1000 képet BGA keresztcsiszolatokról. A képeket felcímkéztem, majd átadtam a mesterséges intelligencia modellnek. A folyamat eredménye a best.pt fájl. Ez a fájl súlyozást, a neurális háló paramétereit tartalmazza. A következőkben ennek a folyamatnak a menetét ismertetem részletesen.

3.5.1 Címkézés

A címkéz fontos lépése a folyamatnak, ettől válik a tanítás felügyelt tanítássá. Itt állítottam elő a bemenethez tartozó helyes kimeneteket. Megadtam a keresendő objektumok helyét és méretét, valamint a hozzájuk tartozó osztályazonosítót.

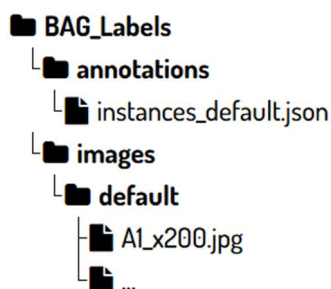
A tanítás során a modell megtanulja a képek és címkék közötti összefüggést, a hibáját a címkéktől való eltérés alapján állapítja meg és módosítja paramétereit, vagyis a súlyokat. Erre a hibakezelésre szolgál a veszteségfüggvény.

3.5.1.1 CVAT (Computer Vision Annotation Tool)

A CVAT - ahogyan azt a neve is mutatja – egy annotáló eszköz. A tanításra szánt képeket egyesével kell felcímkézni, a detektálni kívánt részeket kézzel körbe rajzolni és osztálycímkét adni hozzá. Esetemben 1000 darab képet kellett volna kézzel feljelölnöm, amely rengeteg időt vett volna igénybe. Ennek elkerülése érdekében használtam a CVAT-ot.

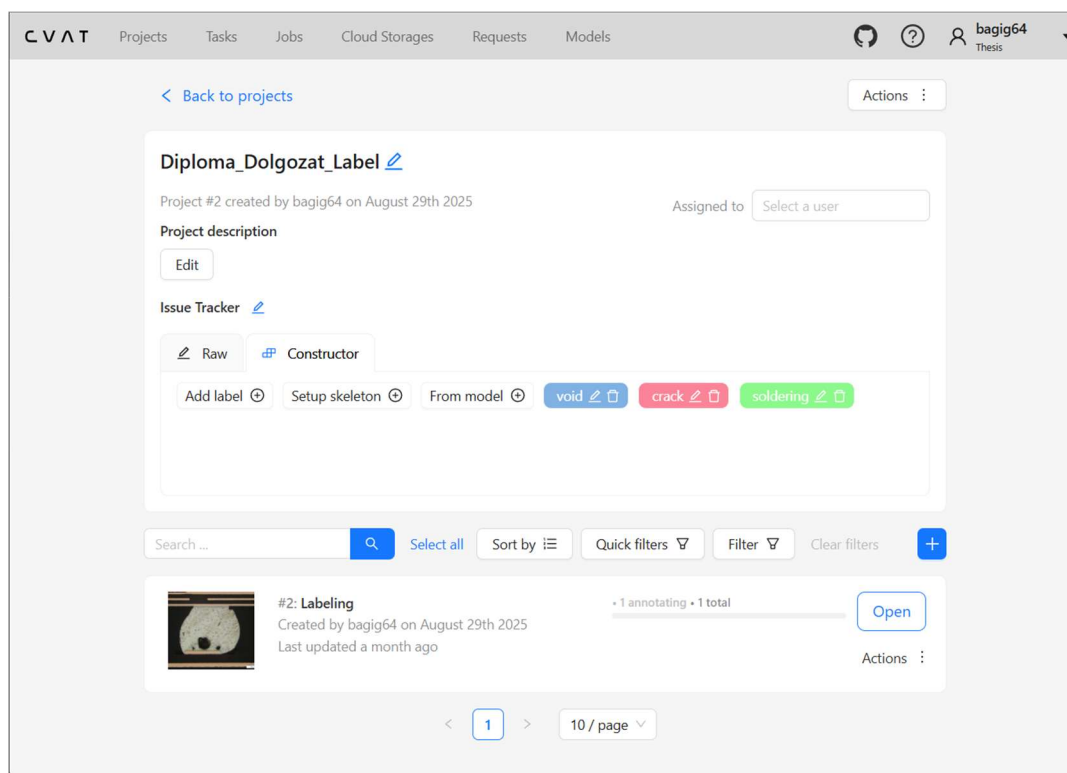
A CVAT-ban előre definiáltam az osztályokat, amiket jelölni szeretnék. A címkézést két alkalommal végeztem el, először a hibákat jelöltem, később pedig a forrasztást. Ennek megfelelően kétszer tanítottam az MI modellt is. Azért választottam ezt a megoldást, hogy a százalékos értékek meghatározásánál külön tudjam kezelni a forrasztást és a hibákat. Első körben tehát két osztályt hoztam létre: void és crack. A feljelöléshez két eszközt használtam, az automata és a kézi ecset jelölőt. Az automatát a void-hoz, az ecsetet a crack-hez.

Ezt követően az automatával elvégeztem a forrasztások feljelölését is, miután előre definiáltam ezt a harmadik osztályt is. Az exportálás eredménye egy tömörített állomány két mappával:

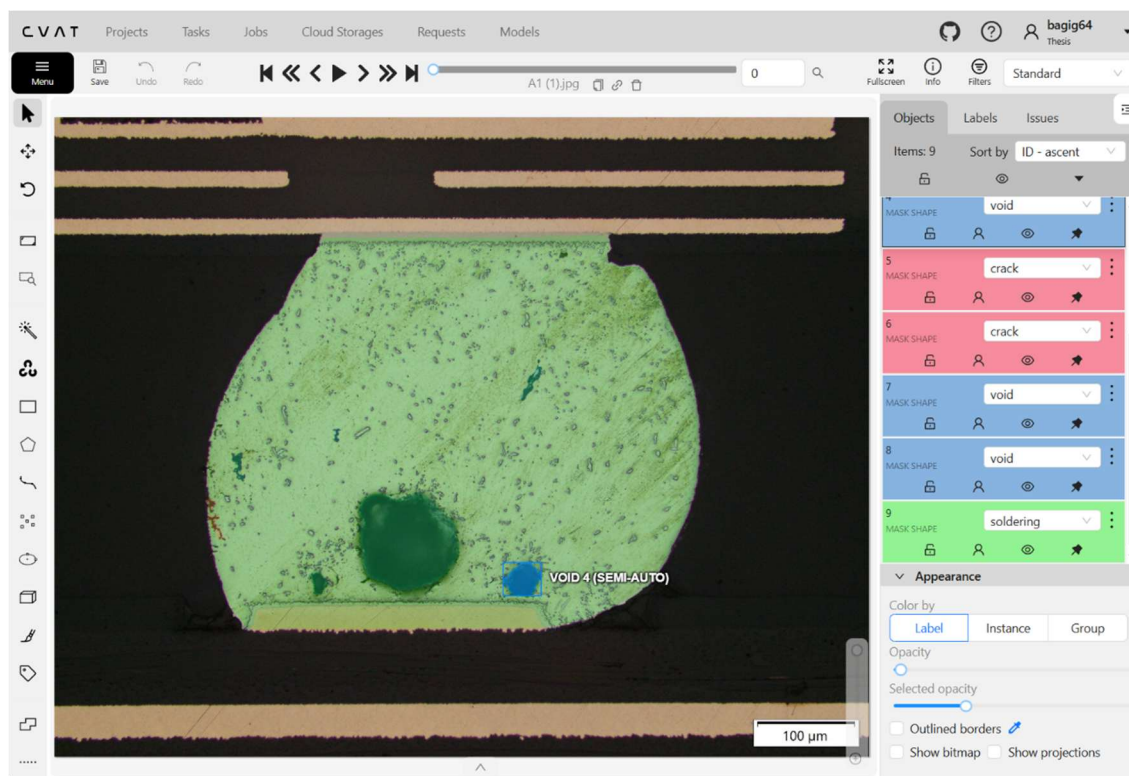


14. ábra Tömörített állomány struktúrája

Az annotations mappa tartalmazza az instances_default.json fájlt, amelyben a címkézések és a címkékhez tartozó osztályok vannak képenként, az images default mappája pedig a képeket tartalmazza. [32] [33]



15. ábra CVAT képernyőkép - Diploma_Dolgozat_Label



16. ábra CVAT képernyőkép annotálási folyamat közben

3.5.1.2 Konvertálás

A modell számára megfelelő bemenet formája a YOLO Format, amelyben a képek elnevezésével és számával megegyező .txt fájlok tárolják a címkézés adatait. Ehhez a konverzióhoz az Ultralytics JSON2YOLO eszközt használtam. [34]

Ezt követően a fájlokat összepárosítottam egy programszkript segítségével, majd az adatok egyharmad részét kijelöltem validációs adatnak. Erre azért volt szükség, hogy a tanítás után a modellt olyan bemenetekkel tudjam tesztelni, amelyeket a tanítás során nem látott.

3.5.2 Tanítás

A tanítás a „yolov8n-seg.pt” alapmodell betöltésével kezdtem. A tanítást átirányítottam a videokártyára (GPU), ezzel lecsökkentve a folyamathoz szükséges időt. Ezt követően indítottam el a tanítást. A tanítás konfigurációs fájljában adtam meg a tanítási adatok mappájának elérési útvonalát, azon belül pedig külön a tanításra és a validálásra szánt adatok mappáit. Itt adtam meg az osztályok számát és azok megnevezését is. Az Ultralytics oldalán

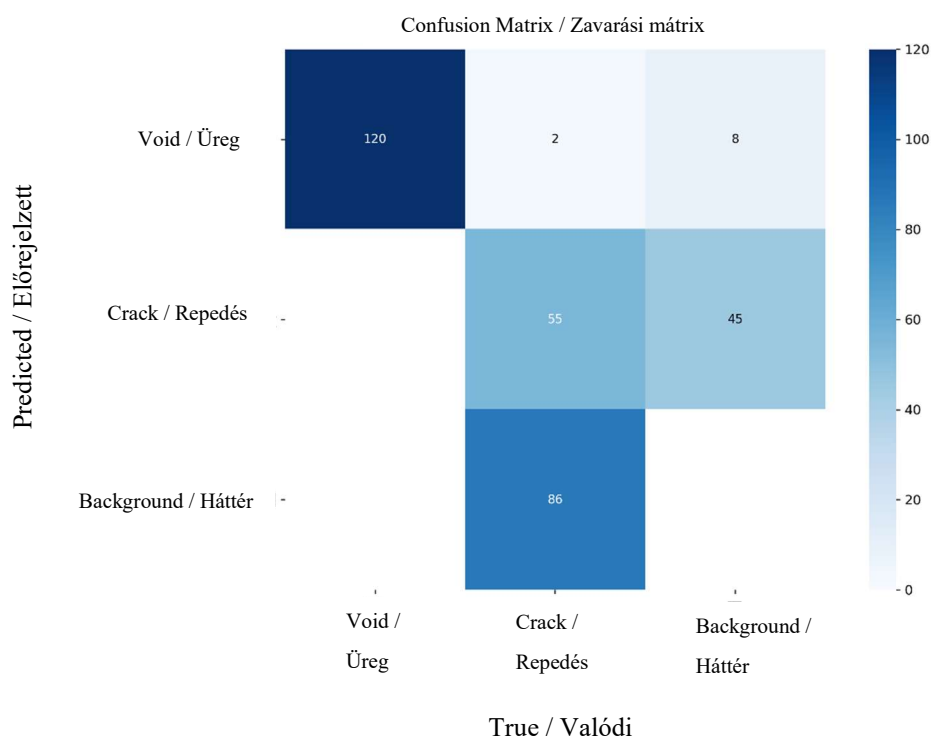
található ajánlást követve az epochs, vagyis a tanítási körök számát 100-ra, a bemeneti képek méretét 640x640 pixelre skálázva adtam meg. [35]

3.5.3 Eredmény

Az eredmények automatikusan jönnek létre a runs mappában. Itt található a weights mappa, amely a best.pt és last.pt fájlokat tartalmazza. A best.pt a validációs adathalmazon legjobban teljesítő modell súlyait, a last.pt az utolsó epoch után mentett súlyokat tartalmazza. Ezek közül én a best.pt-t használtam.

Ezen felül képek és grafikonok is találhatók az eredményben. Ezek segítségével nyomon követhető és értékelhető a tanulás folyamata.

A modell teljesítményének kiértékeléséhez a következő „confusion mátrix” nyújt segítséget.

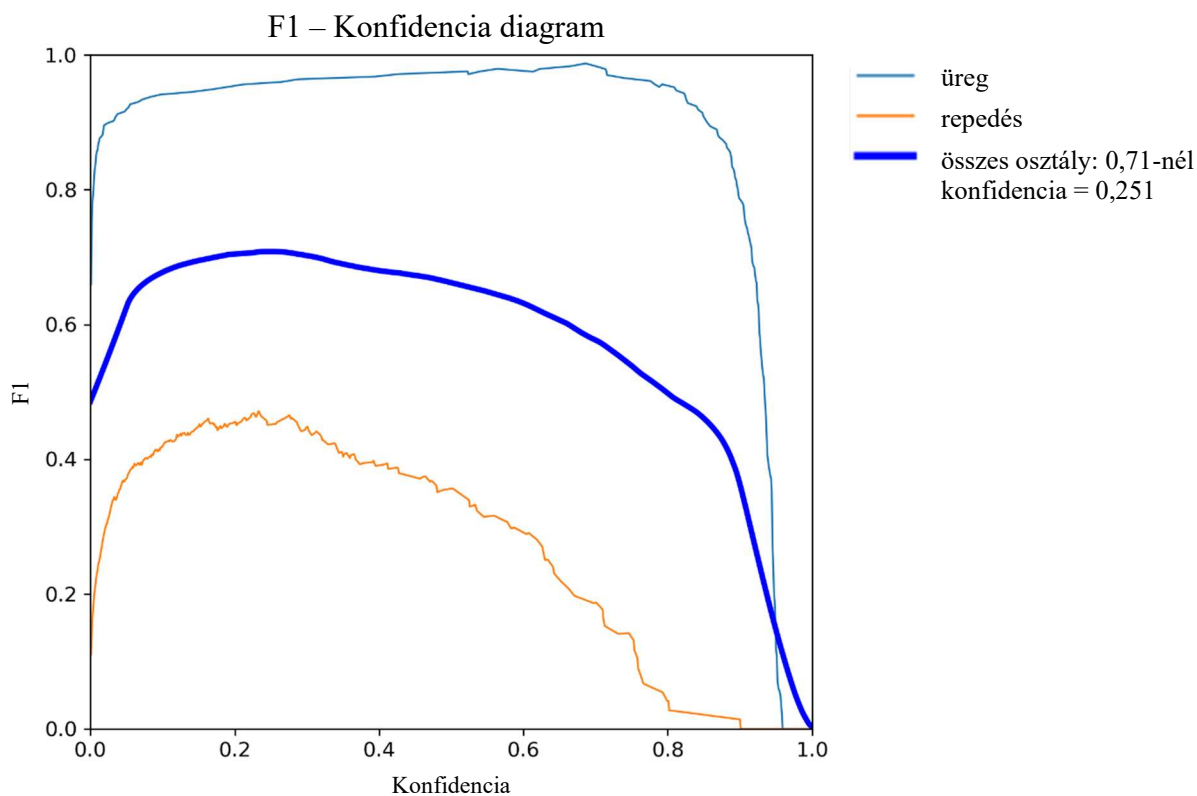


17. ábra Confusion Matrix

A mátrixból meghatározható:

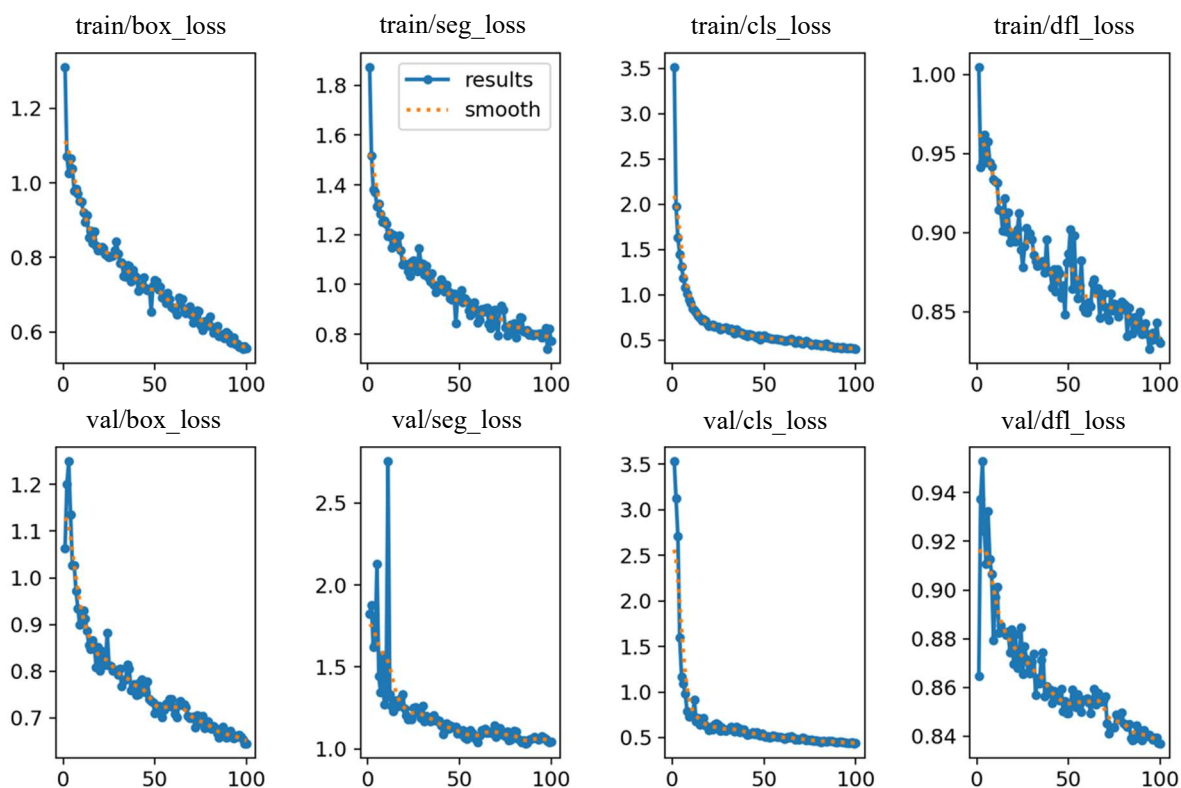
- Pontosság: Mennyire jó összességében a modell?
- Precizitás: Mennyire bízhatunk benne?
- Érzékenység: A találatokból mennyi helyes?
- F1 mutató: A precizitás és az érzékenység kombinációja.

Az ábrán az x-tengely azt jelenti, hogy a modell mennyire biztos a detektálásban, az y-tengelyen lévő F1 érték a modell pontosságáról és érzékenységről hordoz információt. F1 értéke attól függ, hogy hány objektumot talál meg és abból mennyi felel meg, vagy éppen hányat nem talál meg. Esetünkben a diagramból jól látszik, hogy az üregeket szinte hibátlanul megtanulta a modell, azonban a repedés detektálásában jóval gyengébb teljesítményt mutat. Ebben a formában az F1 érték maximuma 0.251 konfidencia értéknél érhető el, ami arra utal, hogy a modell nem túl magabiztos. Ennek javítására érdemes lenne tovább folytatnom a tanítást új címkézett adatokkal. [36]



18. ábra F1-Konfidencia diagram

A veszteségfüggvények exponenciális csökkenést mutatnak, amelyet megfelelőnek ítélek.

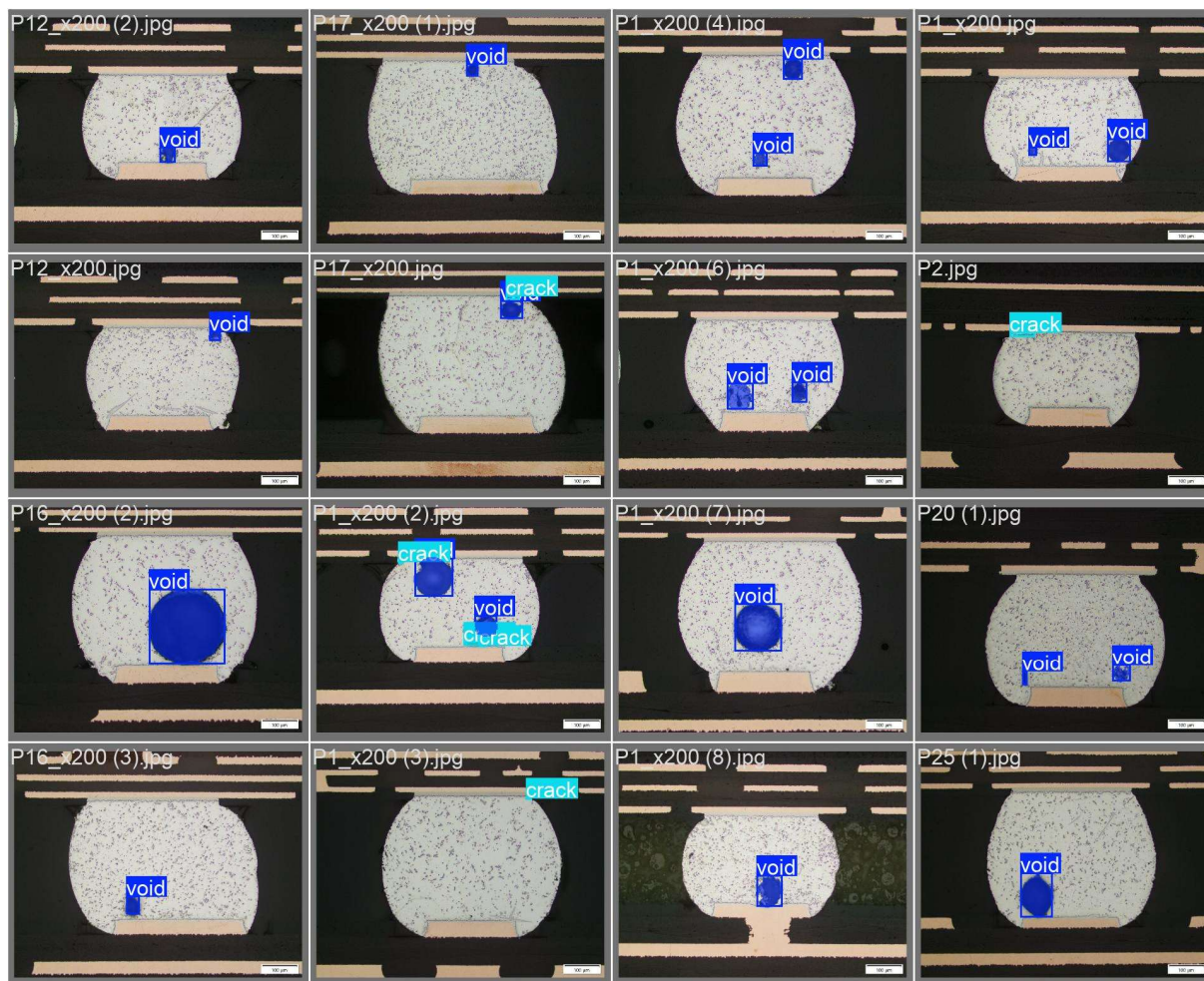


19. ábra Veszteségdiagramok

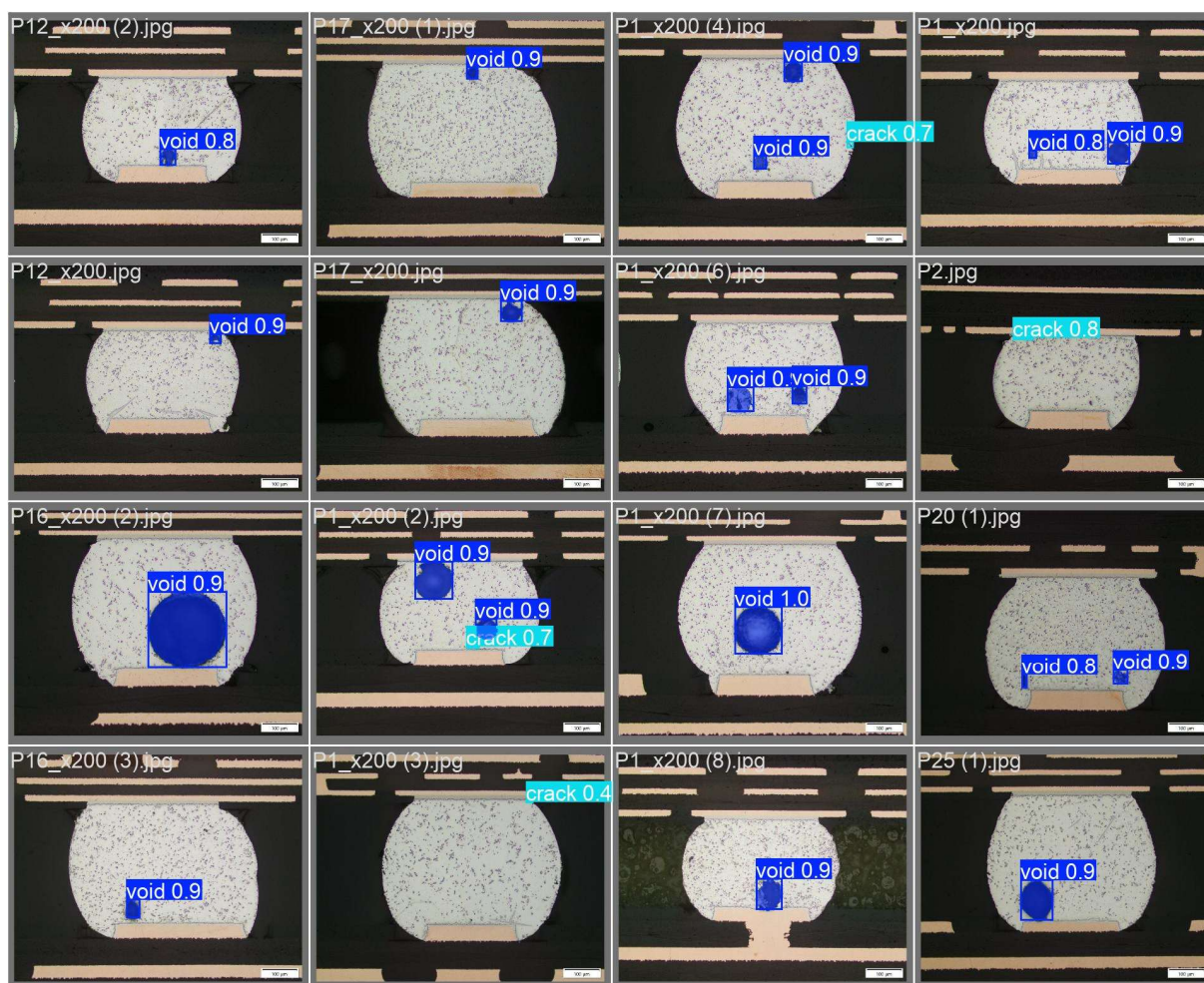
A 19. ábra minden grafikonja a tanítási ciklusok számát ábrázolja a veszteség függvényében, ahol:

- train/box_loss → tanítás/doboz veszteség
- train/seg_loss → tanítás/szegmentációs veszteség
- train/cls_loss → tanítás/osztályozási veszteség
- train/dfl_loss → tanítás/eloszlási veszteség
- val/box_loss → érvényesítés/doboz veszteség
- val/seg_loss → érvényesítés/szegmentációs veszteség
- val/cls_loss → érvényesítés/osztályozási veszteség
- val/dfl_loss → érvényesítés/eloszlási veszteség

Az eredményben megtalálható a validáció, összehasonlítható a címkézés és a jóslás.



20. ábra Kézzel címkézett képek



21. ábra Predikciós eredmény

A két képkötegek összevetve jól látszik, hogy a üregek meghatározása kivétel nélkül sikeresen megtörtént, a repedés meghatározásában azonban már az ennél a néhány képnél is felfedezhető eltérés. [37]

3.5.4 Értékelés

A modell tanítása elsődlegesen a Void – Forraszanyag meghatározására irányult, a Crack tanítása továbbfejlesztési szándékkal került be a modellbe. Ennek okán a modell tanítását sikeresnek értékelem, a Crack bizonytalan detektálása szemléltetés szempontjából pozitív a projektre nézve. [38]

További értékelő diagrammok és képek a Mellékletek fejezetben találhatóak a Soldering tanítási folyamatot illetően is.

4. Gazdasági számítás

Ebben a fejezetben bemutatom a projekthez szükséges anyagi forrást.

A programhoz használt környezetek, eszközök mind ingyenesen elérhetőek. Emiatt tehát az alkalmazás közvetlen költségeinek értéke 0 Ft.

A közvetett költségekbe beletartozik a fejlesztő bére, a felhasznált áram, a hardver amortizációja, amelyen a fejlesztés zajlott. Ezen költségek közül a felhasznált villamos energia ára elhanyagolható, csakúgy, mint az eszköz amortizációja. A számottevő költség a fejlesztő bére. A munkabér meghatározása a magyar mérnöki kamara 2025. január 1-től érvényben lévő mérnöki díjszabás táblázata alapján történik.

Kezdő mérnök esetében az ajánlott mérnöki napidíj: 123.000 Ft [39]

A díjszabás tartalmazza:

- a szolgáltatást végzők bruttó munkadíja, valamint ezek járuléka,
- adók (az általános forgalmi adó kivételével),
- az esetlegesen a szerzői joggal kapcsolatosan felmerülő személyi és vagyoni jogok értéke,
- az értékcsökkenési leírás,
- vállalkozói nyereség,
- általános rezsiköltségek

A projekt elkészítés kettő hetet vesz igénybe, ha a fejlesztő napi 8 órát munkával tölt. Ebbe a hétvégék nem tartoznak bele. [39]

4.1 Összegzés

Költségtípus	Összeg (Ft)	Megjegyzés
Fejlesztői környezet	0,00 Ft	Ingyenes (VS Code, Python)
Annotációs eszköz	0,00 Ft	Ingyenes (CVAT)
Modell (YOLO)	0,00 Ft	Nyílt forráskódú
Összes költség	1 230 000,00 Ft	főleg munkaidő

A projekt összköltsége tehát 1 230 000Ft, amelyet teljes egészében a kezdő mérnök bére tesz ki.

5. Összefoglalás

A dolgozatom témája egy program fejlesztése volt, amely képes BGA (Ball Grid Array) golyós rácsömb típusú nyomtatott áramkör tokozások keresztcsiszolati képén automatizálva meghatározni az üres tér és a forrasztóanyag százalékos arányát, ezzel támogatva az analízist végző személy munkáját.

A projekt megvalósításához mesterséges intelligenciát használtam. Ennek tanításához nagy mennyiségű képre volt szükségem keresztcsiszolatokról. Ezeknek a képeknek egy részéhez a csiszolatokat és a képeket én készítettem inverz metallográf mikroszkóppal 200-szoros nagyításban.

Ezt követően létrehoztam a GUI (Graphical User Interface) grafikus felhasználói felületet az intuitív használathoz. A program megnyitását követően a felhasználó egy üdvözléssel összekötött választófelületre kerül, ahol kiválaszthatja a vizsgálat alapjaként szolgáló keresztcsiszolat típusát. Választható eszköz a BGA, felület- és furatszerelt alkatrészek. Az utóbbi kettő továbbfejlesztési szempontból került a programba, ezeket választva erről a felhasználót tájékoztatom, majd visszairányítom a kezdőoldalra. A BGA oldalon a felhasználó betölthet egy képet, képek egy mappáját, futtathat analízist egy képre vagy egy mappa képeire automatikusan, az eredményeket PDF-ként exportálhatja. Az megnyitott kép az ablak közepén jelenik meg, analízist követően címkékkel kiegészítve. Az ablak jobb oldalán jelenik meg a vizsgált eszköz típusa, csiszolati síkjának sorszáma, illetve a megjelenített forrasztógolyó neve. Az analízis futtatását követően a program dönt a forrasztás megfelelőségéről is.

A mesterséges intelligencia modell implementálásához egy előre tanított YOLO (You Only Look Once) modellt tanítottam át a sajátcélú felhasználáshoz. A tanítás során továbbfejlesztési szándékkal bevitettem a repedés osztályt is. A tanítás és validálási folyamatot követően az eredményekből jól látszik, hogy a modell az üres terek detektálását szinte 100 százalékban megtanulta, a repedések vizsgálatához viszont ajánlott a modell tovább tanítása új adatokkal.

Összeségében a program a vele szembeállított követelményeknek megfelelt, további fejlesztési irány lehet a felület- és furatszerelt alkatrészek forrasztásának ellenőrzése, tovább tanítással a repedések hosszának meghatározása.

6. Summary

The topic of my thesis was the development of a program capable of automatically determining the percentage ratio of voids and solder material in cross-sectional images of BGA (Ball Grid Array) type printed circuit board packages, thereby supporting the work of the analyst.

For the implementation of the project, I used artificial intelligence. Training required a large number of cross-sectional images. Some of these images I created myself, by preparing the cross-sections and capturing the images using an inverted metallographic microscope at 200× magnification.

Subsequently, I developed a GUI (Graphical User Interface) for intuitive use. Upon launching the program, the user is greeted with a welcome screen and a selection interface where they can choose the type of cross-section to be analysed. The available options are BGA, surface-mounted, and through-hole components. The latter two were included as potential future extensions of the program; when selected, the user is notified accordingly and redirected back to the main menu. On the BGA page, the user can load a single image or an entire folder of images, run an analysis on one or multiple images automatically, and export the results as a PDF file. The opened image is displayed in the center of the window and, following analysis, is annotated with labels. On the right-hand side of the window, the program displays the type of component examined, the index of the cross-section plane, and the identifier of the analyzed solder ball. After performing the analysis, the program also makes a judgment on the adequacy of the soldering.

For the implementation of the artificial intelligence model, I retrained a pre-trained YOLO (You Only Look Once) model for this specific application. To enable further development, I also introduced a “crack” class during training. Following the training and validation process, the results showed that the model learned to detect voids with nearly 100% accuracy, while in the case of cracks, additional training with new data is recommended.

In summary, the program met the specified requirements. Potential directions for further development include the inspection of surface-mounted and through-hole solder joints, as well as the determination of crack length through additional training.

7. Nyilatkozatok

MATE Szervezeti és Működési Szabályzat

III. Hallgatói Követelményrendszer

III.1. Tanulmányi és Vizsgaszabályzat

6.13. sz. függelék: A MATE egységes szakdolgozat /

diplomadolgozat / záródolgozat / portfólió készítési útmutatója

4.2. sz. melléklete: Nyilatkozat a záródolgozat/szakdolgozat/diplomadolgozat/portfólió nyilvános hozzáféréseiről és eredetiségéről (módosítva: 2025. október 16.)

NYILATKOZAT

a záródolgozat/szakdolgozat/diplomadolgozat/portfólió¹ nyilvános hozzáféréseiről és eredetiségéről

A hallgató neve:	Bagi Gergő
A Hallgató Neptun kódja:	DZSJ1K
A dolgozat címe:	Mesterséges intelligenciával támogatott anyagvizsgálat
A megjelenés éve:	2025
A konzulens intézetének neve:	Műszaki Intézet
A konzulens tanszékének a neve:	Mechatronika Tanszék

Kijelentem, hogy az általam benyújtott záródolgozat/szakdolgozat/diplomadolgozat/portfólió² egyéni, eredeti jellegű, saját szellemi alkotásom. Azon részeket, melyeket más szerzők munkájából vettem át, egyértelműen megjelöltem, és az irodalomjegyzékben szerepeltettem. Továbbá kijelentem, hogy a dolgozat elkészítése során alkalmazott mesterséges intelligencia-eszközök (pl. szöveggenerálás, nyelvi javítás, fordítás, adatelemzés) használata nem helyettesítette a saját kutatási és alkotói munkámat, azok alkalmazását a források között vagy a módszertani részben feltüntettem, és a szakmai-etikai elvárásoknak megfelelően jártam el.

Ha a fenti nyilatkozattal valótlant állítottam, tudomásul veszem, hogy a záróvizsga-bizottság a záróvizsgából kizár és a záróvizsgát csak új dolgozat készítése után tehetek.

A leadott dolgozat, mely PDF dokumentum, szerkesztését nem, megtekintését és nyomtatását engedélyezem.

Tudomásul veszem, hogy az általam készített dolgozatra, mint szellemi alkotás felhasználására, hasznosítására a Magyar Agrár- és Élettudományi Egyetem mindenkor szellemi tulajdon-kezelési szabályzatában megfogalmazottak érvényesek.

Tudomásul veszem, hogy dolgozatom elektronikus változata feltöltésre kerül a Magyar Agrár- és Élettudományi Egyetem könyvtári repozitóri rendszerébe. Tudomásul veszem, hogy a megvédett és

- nem titkosított dolgozat a védést követően
- titkosításra engedélyezett dolgozat a benyújtásától számított 5 év eltelté után

nyilvánosan elérhető és kereshető lesz az Egyetem könyvtári repozitóri rendszerében.

Kelt: 2025. év 10. hó 26. nap


Hallgató aláírása

¹ A megfelelő dolgozattípus meghagyása mellett a többi típus törlendő.

² A megfelelő dolgozattípus meghagyása mellett a többi típus törlendő.

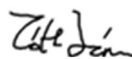
NYILATKOZAT

Bagi Gergő (név) (hallgató Neptun azonosítója: DZSJ1K) konzulenseként nyilatkozom arról, hogy a záródolgozatot/szakdolgozatot/diplomadolgozatot/portfóliót¹ áttekintettem, a hallgatót az irodalmi források korrekt kezelésének követelményeiről, jogi és etikai szabályairól tájékoztattam.

A záródolgozatot/szakdolgozatot/diplomadolgozatot/portfóliót a záróvizsgán történő védésre javaslom / nem javaslom².

A dolgozat állam- vagy szolgálati titkot tartalmaz: igen nem^{*3}

Kelt: 2025 év 10 hó 26 nap



belső konzulens

¹ A megfelelő dolgozattípus meghagyása mellett a többi típus törlendő.

² A megfelelő aláhúzendő.

³ A megfelelő aláhúzendő.

MATE Szervezeti és Működési Szabályzat

III. Hallgatói Követelményrendszer

III.1. Tanulmányi és Vizsgaszabályzat

**6.13. sz. függelék: A MATE egységes szakdolgozat /
diplomadolgozat / záródolgozat / portfólió készítési útmutatója**

7. sz. melléklete: Műszaki Intézet külső konzulensi nyilatkozat

KÜLSŐ KONZULENSI NYILATKOZAT

Bagi Gergő (név) (hallgató Neptun azonosítója: DZSJ1K)

külső konzulenseként nyilatkozom arról, hogy a hallgató az előre egyeztetett konzultációkon rendszeresen megjelent.

Kelt: 2025 év 10 hó 26 nap



külső konzulens

Hallgatók, doktoranduszok nyilatkozata mesterséges intelligencia (MI) alkalmazásáról

1. Általános adatok

Hallgató neve:	Bagi Gergő
Neptun-kódja:	DZSJ1K
Képzési szint (a megfelelőt jelölje X-szel):	☐ BSc/BA ☒ MSc/MA ☐ Doktori (PhD) ☐ Egyéb:
Tantárgy neve/kódja*:	Diplomadolgozat Készítése II. / MUSZK051N
A munka címe:	Mesterséges Intelligenciával Támogatott Anyagvizsgálat

* doktori értekezés esetén nem kitöltendő

2. Nyilatkozat az MI használatáról

Alulírott, etikai felelősségem teljes tudatában az alábbi nyilatkozatot teszem:

(Kérjük, válasszon egyet az alábbi lehetőségek közül!)

☐ A) Nem alkalmaztam mesterséges intelligencia rendszert vagy szolgáltatást.

(Amennyiben ezt jelölte, a további táblázatok kitöltése nem szükséges.)

☒ B) Alkalmaztam mesterséges intelligencia rendszert vagy szolgáltatást.

(Kérjük, töltsse ki a vonatkozó táblázatokat!)

3. A mesterséges intelligencia használatának részletezése

I. TÁBLÁZAT: Asszisztensi vagy kisebb mértékű felhasználás (pl. fordítás, nyelvi korrektúra, ötletelés stb.)

(Ezen felhasználások esetében a konkrét promptok és válaszok csatolása nem szükséges.)

A felhasználás célja	Alkalmazott MI-eszköz neve és verziója	Érintett rész (ha nem a szöveg egészére vonatkozik)
fordítás	ChatGPT, GPT-5	Summary

II. TÁBLÁZAT: Jelentős tartalmi hozzájárulás (pl. egy teljes ábra vagy egy hosszabb szövegrész generálása)

(Ezekben az esetekben a felhasznált kulcsfontosságú promptok és az MI által adott nyers válaszok dokumentálása és a munka mellékletében való csatolása szükséges.)

A felhasználás célja	Alkalmazott eszköz verziója, elérhetősége	MI-neve,	Az érintett fejezet / ábra / táblázat pontos sorszáma	A prompt-naplót tartalmazó melléklet bejegyzésének sorszáma

3/A. Oktató által előírt kiegészítő szabályok (ha vannak)

Amennyiben az adott tantárgy oktatója vagy témavezetője az MI-eszközök használatára vonatkozóan külön szabályokat vagy elvárásokat határozott meg, kérjük, az alábbi mezőben foglalja össze ezeket:

Pl. az MI használatának tilalma bizonyos feladattípusokra; csak konkrét eszköz használata engedélyezett; eltérő hivatkozási elvárások; dokumentációs forma stb.

Oktató vagy témavezető által előírt szabályok:

.....

.....

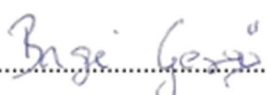
.....

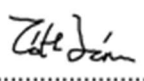
.....

4. Minden hallgatóra vonatkozó nyilatkozat:

Kijelentem, hogy az MI által esetlegesen generált tartalmakat minden esetben kritikailag felülvizsgáltam, szerkesztettem és a munkába illesztettem. A leadott munka minden eleméért, annak eredetiségéért és tudományos helytállóságáért teljes körű felelősséget vállalok. Tudomásul veszem, hogy a Magyar Agrár- és Élettudományi Egyetem a benyújtott munkát mesterséges intelligencia detektorral ellenőrizheti, és eljárást kezdeményezhet, amennyiben a nyilatkozatom valótlan vagy hiányos.

Kelt: Gödöllő 2025. 10 hó 26 nap

.....

Hallgató aláírása

.....

Konzulens/Témavezető aláírása

8. Irodalomjegyzék

- [1] K. Turcsányi, „Minőségelmélet és - módszertan,” Nemzeti Közszerológati Egyetem, Budapest, 2014.
- [2] J. Deákné Kovács, „Anyagvizsgálatok- Roncsolásmentes vizsgálati módszerek 1.,” Nemzeti Szakképzési és Felnőttképzési Intézet, Budapest, 2008.
- [3] D. L. C.P. Wong, Materials for Advanced Packaging, E-könyv: Springer International Publishing, 2016.
- [4] M. Dr. Szalóki, K. Dr. Bukovinszky, A. Dr. Jenei, I. Dr. Lampé és J. Bakó, „Anyagvizsgálati módszerek,” 2015.
- [5] M. Brennan, A Practical Approach to Quantitative Metal Analysis of Organic Matrices, E-könyv: Wiley, 2008.
- [6] L. Cseh-Bognár, „Metallográfiai vizsgálatok,” Atestor Kft., Budapest, 2012.
- [7] G. F. V. Voort, Metallography--past, Present, and Future, Baltimore: ASTM, 1993.
- [8] G. M. S. H. B.D. Sartwell, Metallurgical Coatings and Thin Films, E-könyv: Elsevier Science, 2012.
- [9] G. F. V. Voort, Metallography, Principles and Practice, E-könyv: ASM International, 1999.
- [10] F. Beck, Integrated Circuit Failure Analysis, Chichester: Wiley, 1998.
- [11] J. H. Lau, Ball Grid Array Technology, McGraw-Hill, 1995.
- [12] W. Greig, Integrated Circuit Packaging, Assembly and Interconnections, E-könyv: Books24x7.com, 2007.

-
- [13] S. Russel és P. Norvig, „Mesterséges Intelligencia Modern megközelítésben Második, átdolgozott, bővített kiadás,” Hungarian Translation Panem Könyvkiadó, Budapest, 2005.
- [14] P. K. G. Lavanya Sharma, Artificial Intelligence, E-könyv: CRC Press, 2021.
- [15] S. D. Mohaghegh, Artificial Intelligence for Science and Engineering Applications, E-könyv: CRC Press, 2024.
- [16] A. Jain, „All about convolutions, kernels, features in CNN,” Medium, 2024.
- [17] B. L. Ragav Venkatesan, Convolutional Neural Networks in Visual Computing, E-könyv: CRC Press, 2017.
- [18] F. Millstein, Convolutional Neural Networks In Python, E-könyv: Frank Millstein, 2020.
- [19] H. R. S. A. A. S. M. B. Salman Khan, A Guide to Convolutional Neural Networks for Computer Vision, E-könyv: Springer International Publishing, 2022.
- [20] A. Lambda, *Let's Visualize How YOLO Works*, <https://www.youtube.com/watch?v=BRS9bEP8rhA>, 2024.
- [21] Object Detection with Deep Learning Models, E-könyv: CRC Press, 2022.
- [22] S. Kang, Z. Hu, L. Liu, K. Zhang és Z. Cao, „Object Detection YOLO Algorithms and Their Industrial Applications: Overview and Comparative Analysis,” MDPI (Multidisciplinary Digital Publishing Institute), Basel, Svájc, 2025.
- [23] R. Sapkota, M. Flores-Calero, R. Qureshi, C. Badgujar, U. Nepal, A. Poulouse, P. Zeno, U. Prakash Vaddevolu, S. Khan, M. Shoman, H. Yan és M. Karkee, „YOLO advances to its genesis: a decadal and comprehensive review of the You Only Look Once (YOLO) series,” The Author(s), 2025.
- [24] J. Redmon, S. Divvala, R. Girshick és A. Farhadi, „You Only Look Once: Unified, Real-Time Object Detection,” University of Washington, Allen Institute for AI, Facebook AI Research, New York, 2016.

-
- [25] Reliability Theory and Technology in Manufacturing Process, E-könyv: Springer Nature Singapore, 2025.
- [26] Mckeown, Mechanical Analysis of Electronic Packaging Systems, New York: Taylor & Francis, 1999.
- [27] P. Rowland és M. Richard, „Metallographic Cross Sectioning Techniques for Performance of Destruct Physical Analysis of Glass Encapsulated Diodes,” *MicroNotes*.
- [28] K. A. Katona József, Objektumorientált szoftverfejlesztés alapjai, Publio Kiadó Kft., 2015.
- [29] Python tudásépítés lépésről lépésre az alapoktól az első asztali alkalmazásig, E-könyv: Deuteron Kft., 2025.
- [30] J. M. Willman, Beginning PyQt, E-könyv: Apress, 2020.
- [31] Q. C. Ltd., „Qt for Python Documentation,” 2022. [Online]. Available: <https://doc.qt.io/archives/qtforpython-5/contents.html>. [Hozzáférés dátuma: 20 szeptember 2025].
- [32] C. Team, „CVAT Adds YOLOv8 Format Support for Seamless Dataset Import and Export,” 2024. [Online]. Available: <https://www.cvat.ai/resources/blog/cvat-yolov8-support>. [Hozzáférés dátuma: 20 szeptember 2025].
- [33] V. K. Suda, Data Labeling in Machine Learning with Python, E-könyv: Packt Publishing, 2024.
- [34] Ultralytics, „Github,” 2024. [Online]. Available: <https://github.com/ultralytics/JSON2YOLO>. [Hozzáférés dátuma: 27 09 2025].
- [35] Ultralytics, „Ultralytics YOLO Docs,” 2024. [Online]. Available: <https://docs.ultralytics.com/>. [Hozzáférés dátuma: 20 szeptember 2025].
- [36] K. Elven, „YOLO: How to calculate F1-score at IOU0.5,” Medium, <https://medium.com/@elvenkim1/yolo-how-to-calculate-f1-score-at-iou0-5-8e665d36ffd5>, 2024.

-
- [37] Ultralytics, „Model Training with Ultralytics YOLO,” 2024. [Online]. Available: <https://docs.ultralytics.com/modes/train/>. [Hozzáférés dátuma: 20 szeptember 2025].
- [38] Ultralytics, „Model Validation with Ultralytics YOLO,” 2024. [Online]. Available: <https://docs.ultralytics.com/modes/val/>. [Hozzáférés dátuma: 20 szeptember 2025].
- [39] M. M. Kamara, „Mérnöki ajánlott díjszabás,” [Online]. Available: <https://www.mmk.hu/informaciok/hirek/mernoki-dijszabas-202501>. [Hozzáférés dátuma: 11 10 2025].
- [40] O. Malyshev, „The Use of CNN for Image Processing,” Svitla, 2024.
- [41] V. M. N. S. S. V. K. R. G. V. C. S. R. N. V. Jagan Mohan, Algorithms in Advanced Artificial Intelligence, E-könyv: CRC Press, 2024.

9. Mellékletek

A mellékletek fejezet tartalmazza a teljes programkódot és a tanítás eredményének minden képét és diagrammját.

9.1 Main.py

```
1. import sys
2. from PyQt5.QtWidgets import QApplication
3. from BaseMainWindow import BaseMainWindow
4. from pages.NewWelcomePage_File import WelcomePage
5.
6. def main():
7.     app = QApplication(sys.argv)
8.     main_window = BaseMainWindow()
9.     welcome = WelcomePage(main_window)
10.    main_window.change_page(welcome)
11.    main_window.show()
12.    sys.exit(app.exec_())
13.
14. if __name__ == "__main__":
15.     main()
16.
```

9.2 BaseMainWindow.py

```
1. from PyQt5.QtWidgets import QMainWindow
2. from PyQt5.QtGui import QIcon
3.
4. class BaseMainWindow(QMainWindow):
5.     def __init__(self):
6.         super().__init__()
7.         self.setWindowTitle("Soldering Inspector")
8.         self.setGeometry(700, 200, 1920, 1080)
9.         #self.setFixedSize(1920, 1080)
10.        self.setWindowIcon(QIcon("assets/Icon.png"))
11.        self.setStyleSheet("""
12.            QMainWindow {
13.                background-image: url('assets/background.png');
14.                background-repeat: no-repeat;
15.                background-position: center;
16.                background-attachment: fixed;
17.            }
18.        """)
19.
20.    def change_page(self, widget):
21.        self.setCentralWidget(widget)
22.
23.
```

9.3 WelcomePage_File.py

```
1. import sys
2. from PyQt5.QtWidgets import (
3.     QApplication, QWidget, QLabel, QHBoxLayout, QPushButton, QVBoxLayout,
4.     QGraphicsDropShadowEffect
5. )
```

```

6. from PyQt5.QtGui import QIcon, QFont, QColor
7. from PyQt5.QtCore import Qt, QSize
8. from .NewBGAWindow_File import BGAWindow
9. from .NewSMDWindow_File import SMDWindow
10. from .NewTHDWindow_File import THTWindow
11. from BaseMainWindow import BaseMainWindow
12.
13.
14. class WelcomePage(QWidget):
15.     def __init__(self, main_window: BaseMainWindow):
16.         super().__init__()
17.         self.main_window = main_window
18.         self.setWindowTitle("Soldering Inspector")
19.         self.setGeometry(700, 200, 1920, 1080)
20.         self.setWindowIcon(QIcon("assets/Icon.png"))
21.         self.setStyleSheet("background-color: #12121f;")
22.         self.initUi()
23.
24.     def initUi(self):
25.         vboxMainBody = QVBoxLayout()
26.
27.         # ===== QuestionMark Button =====
28.         QuestionMark = QPushButton()
29.         QuestionMark.setText("?")
30.         QuestionMark.setFont(QFont("Segoe UI", 25))
31.         QuestionMark.setFixedSize(60, 60)
32.         QuestionMark.setToolTip("This is an AI based soldering inspector application.")
33.         QuestionMark.setStyleSheet(self.revolt_button_style())
34.         shadow = QGraphicsDropShadowEffect()
35.         shadow.setBlurRadius(15)
36.         shadow.setOffset(0)
37.         shadow.setColor(QColor(0, 0, 0, 160))
38.         QuestionMark.setGraphicsEffect(shadow)
39.         vboxMainBody.addWidget(QuestionMark, alignment=Qt.AlignRight)
40.
41.         # ===== Welcome Label =====
42.         WelcomeLabel = QLabel("Welcome to the Soldering Inspector!")
43.         WelcomeLabel.setFont(QFont("Segoe UI", 40))
44.         WelcomeLabel.setFixedSize(1500, 200)
45.         WelcomeLabel.setStyleSheet("""
46.             color: #ffffff;
47.             background-color: #1e1e2f;
48.             border-radius: 30px;
49.         """)
50.         WelcomeLabel.setAlignment(Qt.AlignCenter)
51.         label_shadow = QGraphicsDropShadowEffect()
52.         label_shadow.setBlurRadius(20)
53.         label_shadow.setOffset(0)
54.         label_shadow.setColor(QColor(0, 0, 0, 180))
55.         WelcomeLabel.setGraphicsEffect(label_shadow)
56.         vboxMainBody.addWidget(WelcomeLabel, alignment=Qt.AlignCenter)
57.
58.         # ===== Select Label =====
59.         SelectLabel = QLabel("Please select a device type to analyze:")
60.         SelectLabel.setFont(QFont("Segoe UI", 30))
61.         SelectLabel.setFixedSize(1100, 100)
62.         SelectLabel.setStyleSheet("color: #ffffff; background-color: transparent;")
63.         SelectLabel.setAlignment(Qt.AlignCenter)
64.         vboxMainBody.addWidget(SelectLabel, alignment=Qt.AlignCenter)
65.
66.         # ===== Device Buttons =====
67.         BGAButton = self.create_device_button("BGA \nBall Grid Array", "assets/BGA.png")
68.         SMDButton = self.create_device_button("SMD \nSurface-Mount Device",
69.         "assets/SMD.png")
70.         THTButton = self.create_device_button("THD \nTrough-Hole Device", "assets/THD.png")

```

```

70.
71.         BGAButton.clicked.connect(self.BGA_on_click)
72.         SMDButton.clicked.connect(self.SMD_on_click)
73.         THTButton.clicked.connect(self.THT_on_click)
74.
75.         hbox = QHBoxLayout()
76.         hbox.addStretch()
77.         hbox.addWidget(BGAButton)
78.         hbox.addWidget(SMDButton)
79.         hbox.addWidget(THTButton)
80.         hbox.addStretch()
81.
82.         vboxMainBody.addLayout(hbox)
83.         self.setLayout(vboxMainBody)
84.
85.         # ===== Revolt button style =====
86.         def revolt_button_style(self):
87.             return """
88.             QPushButton {
89.                 background-color: #1e1e2f;
90.                 color: #ffffff;
91.                 border: none;
92.                 border-radius: 15px;
93.                 padding: 15px;
94.                 font-weight: normal;
95.                 qproperty-toolButtonStyle: 3; /* Text under icon */
96.                 transition: all 0.2s;
97.             }
98.             QPushButton: hover {
99.                 background-color: #4f46e5;
100.                 color: #ffffff;
101.             }
102.             QPushButton: pressed {
103.                 background-color: #37364f;
104.                 transform: translateY(2px);
105.             }
106.             """
107.
108.         # ===== Device button creation =====
109.         def create_device_button(self, text, icon_path):
110.             btn = QPushButton()
111.             btn.setText(text)
112.             btn.setFont(QFont("Segoe UI", 25))
113.             btn.setIcon(QIcon(icon_path))
114.             btn.setIconSize(QSize(150, 150))
115.             btn.setToolButtonStyle(Qt.ToolButtonTextUnderIcon)
116.             btn.setStyleSheet(self.revolt_button_style())
117.             shadow = QGraphicsDropShadowEffect()
118.             shadow.setBlurRadius(20)
119.             shadow.setOffset(0, 5)
120.             shadow.setColor(QColor(0, 0, 0, 180))
121.             btn.setGraphicsEffect(shadow)
122.             return btn
123.
124.         # ===== Button actions =====
125.         def THT_on_click(self):
126.             self.main_window.change_page(THTWindow(self.main_window))
127.
128.         def BGA_on_click(self):
129.             self.main_window.change_page(BGAWindow(self.main_window))
130.
131.         def SMD_on_click(self):
132.             self.main_window.change_page(SMDWindow(self.main_window))
133.

```

9.4 SMDWindow_File.py

```

1. from PyQt5.QtWidgets import QWidget, QLabel, QPushButton, QVBoxLayout,
   QGraphicsDropShadowEffect
2. from PyQt5.QtGui import QIcon, QFont, QColor
3. from PyQt5.QtCore import Qt, QSize
4. from BaseMainWindow import BaseMainWindow
5.
6. class SMDWindow(QWidget):
7.     def __init__(self, main_window: BaseMainWindow):
8.         super().__init__()
9.         self.main_window = main_window
10.        self.setWindowTitle("SMD Window")
11.        self.setGeometry(700, 200, 1920, 1080)
12.        self.setWindowIcon(QIcon("assets/Icon.png"))
13.        self.setStyleSheet("background-color: #12121f;")
14.        self.initUi()
15.
16.    def initUi(self):
17.        vboxMainBody = QVBoxLayout()
18.        vboxMainBody.setSpacing(30)
19.
20.        # ===== Title Label =====
21.        title_label = QLabel("This is the SMD window")
22.        title_label.setFont(QFont("Segoe UI", 40))
23.        title_label.setFixedSize(1500, 200)
24.        title_label.setStyleSheet("""
25.            color: #ffffff;
26.            background-color: #1e1e2f;
27.            border-radius: 30px;
28.        """)
29.        title_shadow = QGraphicsDropShadowEffect()
30.        title_shadow.setBlurRadius(20)
31.        title_shadow.setOffset(0)
32.        title_shadow.setColor(QColor(0, 0, 0, 180))
33.        title_label.setGraphicsEffect(title_shadow)
34.        title_label.setAlignment(Qt.AlignCenter)
35.        vboxMainBody.addWidget(title_label, alignment=Qt.AlignCenter)
36.
37.        # ===== Under Development Label =====
38.        dev_label = QLabel(
39.            "This function is under development!\nThank you for your patience!\nCome back
later!"
40.        )
41.        dev_label.setFont(QFont("Segoe UI", 25))
42.        dev_label.setFixedSize(1100, 200)
43.        dev_label.setStyleSheet("color: #ffffff; background-color: transparent;")
44.        dev_label.setAlignment(Qt.AlignCenter)
45.        vboxMainBody.addWidget(dev_label, alignment=Qt.AlignCenter)
46.
47.        # ===== Back to Welcome Button =====
48.        back_button = self.create_revolt_button(
49.            "Back to the Welcome Page",
50.            icon_path="assets/Icon.png",
51.            icon_size=QSize(400, 400),
52.            btn_size=QSize(700, 500)
53.        )
54.        back_button.clicked.connect(self.back_to_welcome_page)
55.        vboxMainBody.addWidget(back_button, alignment=Qt.AlignCenter)
56.
57.        self.setLayout(vboxMainBody)
58.
59.        # ===== Revolt button style =====
60.        def revolt_button_style(self):
61.            return ""

```

```

62.         QPushButton {
63.             background-color: #1e1e2f;
64.             color: #ffffff;
65.             border: none;
66.             border-radius: 20px;
67.             padding: 15px;
68.             font-weight: normal;
69.             qproperty-toolButtonStyle: 3; /* Text under icon */
70.         }
71.         QPushButton:hover {
72.             background-color: #4f46e5;
73.             color: #ffffff;
74.         }
75.         QPushButton:pressed {
76.             background-color: #37364f;
77.             transform: translateY(2px);
78.         }
79.         """"
80.
81.         # ===== Button creation helper =====
82.         def create_revolt_button(self, text, icon_path=None, icon_size=QSize(150,150),
btn_size=QSize(300,300)):
83.             btn = QPushButton()
84.             btn.setText(text)
85.             btn.setFont(QFont("Segoe UI", 25))
86.             btn.setFixedSize(btn_size)
87.             if icon_path:
88.                 btn.setIcon(QIcon(icon_path))
89.                 btn.setIconSize(icon_size)
90.             btn.setToolButtonStyle(Qt.ToolButtonTextUnderIcon)
91.             btn.setStyleSheet(self.revolt_button_style())
92.             shadow = QGraphicsDropShadowEffect()
93.             shadow.setBlurRadius(20)
94.             shadow.setOffset(0, 5)
95.             shadow.setColor(QColor(0, 0, 0, 180))
96.             btn.setGraphicsEffect(shadow)
97.             return btn
98.
99.         # ===== Button action =====
100.        def back_to_welcome_page(self):
101.            from pages.NewWelcomePage_File import WelcomePage
102.            print("Back button clicked")
103.            self.main_window.change_page(WelcomePage(self.main_window))
104.
105.

```

9.5 THDWindow_File.py

```

1. from PyQt5.QtWidgets import QWidget, QLabel, QPushButton, QVBoxLayout,
QGraphicsDropShadowEffect
2. from PyQt5.QtGui import QIcon, QFont, QColor
3. from PyQt5.QtCore import Qt, QSize
4. from BaseMainWindow import BaseMainWindow
5.
6. class THDWindow(QWidget):
7.     def __init__(self, main_window: BaseMainWindow):
8.         super().__init__()
9.         self.main_window = main_window
10.        self.setWindowTitle("THD Window")
11.        self.setGeometry(700, 200, 1920, 1080)
12.        self.setWindowIcon(QIcon("assets/Icon.png"))
13.        self.setStyleSheet("background-color: #12121f;")
14.        self.initUi()

```

```

15.
16.     def initUi(self):
17.         vboxMainBody = QVBoxLayout()
18.         vboxMainBody.setSpacing(30)
19.
20.         # ===== Title Label =====
21.         title_label = QLabel("This is the THD Window")
22.         title_label.setFont(QFont("Segoe UI", 40))
23.         title_label.setFixedSize(1500, 200)
24.         title_label.setStyleSheet("""
25.             color: #ffffff;
26.             background-color: #1e1e2f;
27.             border-radius: 30px;
28.         """)
29.         title_shadow = QGraphicsDropShadowEffect()
30.         title_shadow.setBlurRadius(20)
31.         title_shadow.setOffset(0)
32.         title_shadow.setColor(QColor(0, 0, 0, 180))
33.         title_label.setGraphicsEffect(title_shadow)
34.         title_label.setAlignment(Qt.AlignCenter)
35.         vboxMainBody.addWidget(title_label, alignment=Qt.AlignCenter)
36.
37.         # ===== Under Development Label =====
38.         dev_label = QLabel(
39.             "This function is under development!\nThank you for your patience!\nCome back
later!"
40.         )
41.         dev_label.setFont(QFont("Segoe UI", 25))
42.         dev_label.setFixedSize(1100, 200)
43.         dev_label.setStyleSheet("color: #ffffff; background-color: transparent;")
44.         dev_label.setAlignment(Qt.AlignCenter)
45.         vboxMainBody.addWidget(dev_label, alignment=Qt.AlignCenter)
46.
47.         # ===== Back to Welcome Button =====
48.         back_button = self.create_revolt_button(
49.             "Back to the Welcome Page",
50.             icon_path="assets/Icon.png",
51.             icon_size=QSize(400, 400),
52.             btn_size=QSize(700, 500)
53.         )
54.         back_button.clicked.connect(self.back_to_welcome_page)
55.         vboxMainBody.addWidget(back_button, alignment=Qt.AlignCenter)
56.
57.         self.setLayout(vboxMainBody)
58.
59.         # ===== Revolt button style =====
60.         def revolt_button_style(self):
61.             return """
62.             QPushButton {
63.                 background-color: #1e1e2f;
64.                 color: #ffffff;
65.                 border: none;
66.                 border-radius: 20px;
67.                 padding: 15px;
68.                 font-weight: normal;
69.                 qproperty-toolButtonStyle: 3; /* Text under icon */
70.             }
71.             QPushButton:hover {
72.                 background-color: #4f46e5;
73.                 color: #ffffff;
74.             }
75.             QPushButton:pressed {
76.                 background-color: #37364f;
77.                 transform: translateY(2px);
78.             }

```

```

79.         """
80.
81.         # ===== Button creation helper =====
82.         def create_revolt_button(self, text, icon_path=None, icon_size=QSize(150,150),
btn_size=QSize(300,300)):
83.             btn = QPushButton()
84.             btn.setText(text)
85.             btn.setFont(QFont("Segoe UI", 25))
86.             btn.setFixedSize(btn_size)
87.             if icon_path:
88.                 btn.setIcon(QIcon(icon_path))
89.                 btn.setIconSize(icon_size)
90.             btn.setToolButtonStyle(Qt.ToolButtonTextUnderIcon)
91.             btn.setStyleSheet(self.revolt_button_style())
92.             shadow = QGraphicsDropShadowEffect()
93.             shadow.setBlurRadius(20)
94.             shadow.setOffset(0, 5)
95.             shadow.setColor(QColor(0, 0, 0, 180))
96.             btn.setGraphicsEffect(shadow)
97.             return btn
98.
99.         # ===== Button action =====
100.        def back_to_welcome_page(self):
101.            from pages.NewWelcomePage_File import WelcomePage
102.            print("Back button clicked")
103.            self.main_window.change_page(WelcomePage(self.main_window))
104.
105.

```

9.6 BGAWindow_File.py

```

1. import json
2. import os
3. from PyQt5.QtWidgets import QLabel, QPushButton, QWidget, QVBoxLayout, QHBoxLayout,
QGroupBox, QFrame, QSizePolicy, QFileDialog, QInputDialog, QApplication, QProgressBar
4. from PyQt5.QtGui import QIcon, QFont, QPixmap, QImage
5. from PyQt5.QtCore import Qt
6. import cv2
7. import numpy as np
8. from ultralytics import YOLO
9. from reportlab.lib.pagesizes import A4
10. from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer, Image, Table,
TableStyle
11. from reportlab.lib.styles import getSampleStyleSheet
12. from reportlab.lib import colors
13. from reportlab.platypus import PageBreak
14. from reportlab.lib.units import cm
15.
16. from PyQt5.QtWidgets import (
17.     QLabel, QPushButton, QWidget, QVBoxLayout, QHBoxLayout,
18.     QGroupBox, QProgressBar, QFileDialog, QInputDialog,
19.     QGraphicsDropShadowEffect, QApplication
20. )
21. from PyQt5.QtGui import QIcon, QFont, QPixmap, QColor, QImage
22. from PyQt5.QtCore import Qt, QSize
23.
24. class AnimatedButton(QPushButton):
25.     """Revolt-stílusú gomb animációval hover és click effektekkel"""
26.     def __init__(self, text="", parent=None):
27.         super().__init__(text, parent)
28.         self.setFont(QFont("Segoe UI", 14))
29.         self.setFixedHeight(60)
30.         self.setStyleSheet(self.default_style())

```

```

31.         # Árnyék
32.         shadow = QGraphicsDropShadowEffect()
33.         shadow.setBlurRadius(15)
34.         shadow.setOffset(0, 5)
35.         shadow.setColor(QColor(0,0,0,180))
36.         self.setGraphicsEffect(shadow)
37.
38.     def default_style(self):
39.         return """
40.         QPushButton {
41.             background-color: #1e1e2f;
42.             color: #ffffff;
43.             border: none;
44.             border-radius: 15px;
45.             font-weight: bold;
46.             padding: 10px;
47.         }
48.         QPushButton:hover {
49.             background-color: #4f46e5;
50.             color: #ffffff;
51.         }
52.         QPushButton:pressed {
53.             background-color: #37364f;
54.         }
55.         """
56.
57. class BGAWindow(QWidget):
58.     def __init__(self, main_window):
59.         super().__init__()
60.         self.main_window = main_window
61.         self.setStyleSheet("background-color: #12121f;")
62.         self.image_files = []
63.         self.current_image_index = 0
64.         self.analysis_results = []
65.
66.         # YOLO modellek
67.         self.yolo_model_deviation = YOLO("D:\\MSC_DIPDOLG_Test\\my-python-
project\\MSC_Dipdolg\\ai_files\\deviation\\best.pt")
68.         self.yolo_model_solder = YOLO("D:\\MSC_DIPDOLG_Test\\my-python-
project\\MSC_Dipdolg\\ai_files\\solder\\best.pt")
69.
70.         self.initUi()
71.
72.     def initUi(self):
73.         main_layout = QHBoxLayout(self)
74.         font_family = "Segoe UI"
75.
76.         # ----- Bal panel -----
77.         left_panel = QVBoxLayout()
78.         left_panel.setSpacing(20)
79.
80.         self.load_image_btn = AnimatedButton("Load Image 📁")
81.         self.load_image_btn.clicked.connect(self.load_image)
82.         left_panel.addWidget(self.load_image_btn)
83.
84.         self.load_folder_btn = AnimatedButton("Load Image Folder 📁")
85.         self.load_folder_btn.clicked.connect(self.load_image_folder)
86.         left_panel.addWidget(self.load_folder_btn)
87.
88.         self.run_analysis_btn = AnimatedButton("Run Analysis 📊")
89.         self.run_analysis_btn.clicked.connect(self.run_analysis)
90.         left_panel.addWidget(self.run_analysis_btn)
91.
92.         self.auto_btn = AnimatedButton("Auto-Process ⚙️")
93.         self.auto_btn.clicked.connect(self.run_batch_analysis)

```

```

94.         left_panel.addWidget(self.auto_btn)
95.
96.         self.save_results_btn = AnimatedButton("Save Results )
97.         self.save_results_btn.clicked.connect(self.on_save_results_clicked)
98.         left_panel.addWidget(self.save_results_btn)
99.
100.        left_panel.addStretch()
101.
102.        # ----- Középső panel -----
103.        center_panel = QVBoxLayout()
104.        self.image_label = QLabel("Image Area")
105.        self.image_label.setAlignment(Qt.AlignCenter)
106.        self.image_label.setFixedSize(1000, 800)
107.        self.image_label.setStyleSheet("""
108.            background-color: transparent;
109.            border: 2px solid #333;
110.        """)
111.        center_panel.addWidget(self.image_label, alignment=Qt.AlignCenter)
112.
113.        nav_layout = QHBoxLayout()
114.        self.prev_btn = AnimatedButton(" Previous")
115.        self.prev_btn.clicked.connect(self.show_prev_image)
116.        nav_layout.addWidget(self.prev_btn)
117.
118.        self.next_btn = AnimatedButton("Next )
119.        self.next_btn.clicked.connect(self.show_next_image)
120.        nav_layout.addWidget(self.next_btn)
121.
122.        center_panel.addLayout(nav_layout)
123.
124.        self.thumbnail = QLabel("Thumbnail / Progress")
125.        self.thumbnail.setAlignment(Qt.AlignCenter)
126.        self.thumbnail.setFixedSize(600, 200)
127.        self.thumbnail.setStyleSheet("""
128.            background-color: transparent;
129.            color: #ffffff;
130.        """)
131.        center_panel.addWidget(self.thumbnail, alignment=Qt.AlignCenter)
132.
133.        # ----- Jobb panel -----
134.        right_panel = QVBoxLayout()
135.        right_panel.setSpacing(20)
136.
137.        # Címsorok és betűtípus
138.        font_family = "Segoe UI"
139.
140.        # ===== Statistics box =====
141.        stats_box = QGroupBox("Statistics")
142.        stats_box.setFont(QFont(font_family, 22))
143.        stats_box.setStyleSheet(f"""
144.            QGroupBox {{
145.                color: #ffffff;
146.                background-color: #1e1e2f; /* panel háttér */
147.                border-radius: 20px;
148.                padding: 30px 15px 15px 15px;
149.            }}
150.            QGroupBox:title {{
151.                subcontrol-origin: margin;
152.                subcontrol-position: top center;
153.                background-color: transparent; /* cím háttér */
154.                padding: 0 3px;
155.                color: #ffffff;
156.            }}
157.        """)
158.        # Árnyék hozzáadása

```

```

159.     shadow = QGraphicsDropShadowEffect()
160.     shadow.setBlurRadius(25)
161.     shadow.setXOffset(0)
162.     shadow.setYOffset(5)
163.     shadow.setColor(QColor(0, 0, 0, 160))
164.     stats_box.setGraphicsEffect(shadow)
165.
166.     stats_layout = QVBoxLayout()
167.     stats_layout.setSpacing(20)
168.     self.solder_label = QLabel("Solder: 0.00%")
169.     self.solder_label.setFont(QFont(font_family, 22))
170.     self.solder_label.setStyleSheet("color: #ffffff;" \
171.     "background-color: transparent;")
172.     stats_layout.addWidget(self.solder_label)
173.     self.void_label = QLabel("Void: 0.00%")
174.     self.void_label.setFont(QFont(font_family, 22))
175.     self.void_label.setStyleSheet("color: #ffffff;" \
176.     "background-color: transparent;")
177.     stats_layout.addWidget(self.void_label)
178.     stats_box.setLayout(stats_layout)
179.     stats_box.setFixedHeight(300)
180.     right_panel.addWidget(stats_box)
181.
182.     # ===== Ball box =====
183.     ball_box = QGroupBox("Ball")
184.     ball_box.setFont(QFont(font_family, 22))
185.     ball_box.setStyleSheet(f"""
186.         QGroupBox {{
187.             color: #ffffff;
188.             background-color: #1e1e2f;
189.             border-radius: 20px;
190.             padding: 70px 15px 15px 15px;
191.         }}
192.         QGroupBox:title {{
193.             subcontrol-origin: margin;
194.             subcontrol-position: top center;
195.             background-color: transparent; /* cím háttér */
196.             padding: 10px 3px 20px 3px;
197.             color: #ffffff;
198.         }}
199.     """)
200.
201.     shadow_ball = QGraphicsDropShadowEffect()
202.     shadow_ball.setBlurRadius(25)
203.     shadow_ball.setXOffset(0)
204.     shadow_ball.setYOffset(5)
205.     shadow_ball.setColor(QColor(0, 0, 0, 160))
206.     ball_box.setGraphicsEffect(shadow_ball)
207.
208.     ball_layout = QVBoxLayout()
209.     ball_layout.setSpacing(20)
210.     self.IC_type_label = QLabel("IC type: IC0000")
211.     self.IC_type_label.setFont(QFont(font_family, 22))
212.     self.IC_type_label.setStyleSheet("color: #ffffff;" \
213.     "background-color: transparent;")
214.     ball_layout.addWidget(self.IC_type_label)
215.     self.plane_section_label = QLabel("Plane section: P0")
216.     self.plane_section_label.setFont(QFont(font_family, 22))
217.     self.plane_section_label.setStyleSheet("color: #ffffff;" \
218.     "background-color: transparent;")
219.     ball_layout.addWidget(self.plane_section_label)
220.     self.position_label = QLabel("Position: X1")
221.     self.position_label.setFont(QFont(font_family, 22))
222.     self.position_label.setStyleSheet("color: #ffffff;" \
223.     "background-color: transparent;")

```

```

224.         ball_layout.addWidget(self.position_label)
225.         self.pass_label = QLabel("Pass/Fail")
226.         self.pass_label.setFont(QFont(font_family, 26))
227.         self.pass_label.setStyleSheet("color: yellow; font-weight: bold; background-color:
transparent;")
228.         ball_layout.addWidget(self.pass_label)
229.         ball_box.setLayout(ball_layout)
230.         ball_box.setFixedHeight(350)
231.         right_panel.addWidget(ball_box)
232.         right_panel.addStretch()
233.
234.         # ----- Layout összerakás -----
235.         main_layout.addLayout(left_panel, stretch=1)
236.         main_layout.addLayout(center_panel, stretch=3)
237.         main_layout.addLayout(right_panel, stretch=1)
238.         self.setLayout(main_layout)
239.
240.     def load_image(self):
241.         file_dialog = QFileDialog()
242.         file_path, _ = file_dialog.getOpenFileName(self, "Open Image", "", "Images (*.png
*.jpg *.bmp)")
243.
244.         if file_path:
245.             self.image_files = [file_path]
246.             self.current_image_index = 0
247.
248.             pixmap = QPixmap(file_path)
249.             self.image_label.setStyleSheet("background-color: transparent;")
250.             self.image_label.setPixmap(pixmap.scaled(self.image_label.size(),
Qt.KeepAspectRatio, Qt.SmoothTransformation))
251.             file_name = os.path.basename(file_path)
252.             name_without_extension = os.path.splitext(file_name)[0]
253.             name_without_extension = name_without_extension.split("_")[0] if "_" in
name_without_extension else name_without_extension
254.
255.             self.position_label.setText(f"Position: {name_without_extension}")
256.             self.IC_type_label.setText(f"IC type: Unknown")
257.             self.plane_section_label.setText(f"Plane section: Unknown")
258.             self.next_btn.setEnabled(False)
259.             self.prev_btn.setEnabled(False)
260.
261.     def load_image_folder(self):
262.         folder_path = QFileDialog.getExistingDirectory(self, "Select Image Folder")
263.
264.         if folder_path:
265.             image_extensions = (".png", ".jpg", ".jpeg", ".bmp")
266.             self.image_files = [
267.                 os.path.join(folder_path, f)
268.                 for f in os.listdir(folder_path)
269.                 if f.lower().endswith(image_extensions)
270.             ]
271.             self.image_files.sort()
272.
273.             if self.image_files:
274.                 self.current_image_index = 0
275.                 self.show_image(self.image_files[self.current_image_index])
276.                 self.prev_btn.setEnabled(True)
277.                 self.next_btn.setEnabled(True)
278.
279.     def show_image(self, image_path):
280.         pixmap = QPixmap(image_path)
281.         self.image_label.setPixmap(
282.             pixmap.scaled(self.image_label.size(), Qt.KeepAspectRatio,
Qt.SmoothTransformation)
283.         )

```

```

284.         self.image_label.setStyleSheet("background-color: transparent;")
285.
286.         # Fájlnév alapú címzés a stat panelen
287.         folder_path = os.path.dirname(image_path)
288.         folder_path_name = os.path.basename(folder_path)
289.         if folder_path_name.startswith("IC"):
290.             plane_name = folder_path_name.split("_")[-1]
291.             self.plane_section_label.setText(f"Plane section: {plane_name}")
292.         else:
293.             self.plane_section_label.setText(f"Plane section: {folder_path_name}")
294.
295.         parent_folder_path = os.path.dirname(os.path.dirname(image_path))
296.         parent_folder = os.path.basename(parent_folder_path)
297.
298.         self.IC_type_label.setText(f"IC type: {parent_folder}")
299.
300.
301.
302.         filename = os.path.splitext(os.path.basename(image_path))[0]
303.         filename = filename.split("_")[0] if "_" in filename else filename
304.         self.position_label.setText(f"Position: {filename}")
305.
306.     def show_next_image(self):
307.         if hasattr(self, 'image_files') and self.image_files:
308.             self.current_image_index = (self.current_image_index + 1) %
len(self.image_files)
309.             self.show_image(self.image_files[self.current_image_index])
310.             self.pass_label.setText("-")
311.             self.pass_label.setStyleSheet("color: #ffffff; font-weight: bold; background-
color: transparent;")
312.
313.     def show_prev_image(self):
314.         if hasattr(self, 'image_files') and self.image_files:
315.             self.current_image_index = (self.current_image_index - 1) %
len(self.image_files)
316.             self.show_image(self.image_files[self.current_image_index])
317.             self.pass_label.setText("-")
318.             self.pass_label.setStyleSheet("color: #ffffff; font-weight: bold; background-
color: transparent;")
319.
320.     def run_analysis(self):
321.         # Check if there are images
322.         if not self.image_files:
323.             print("No image loaded to analyze.")
324.             return
325.
326.         image_path = self.image_files[self.current_image_index]
327.
328.         # Load original image (BGR)
329.         original_img = cv2.imread(image_path)
330.
331.         if original_img is None:
332.             print("Failed to load image.")
333.             return
334.
335.         # --- Run first model (deviation) ---
336.         results_dev = self.yolo_model_deviation(image_path)
337.         annotated_img = results_dev[0].plot() # returns a numpy array (BGR)
338.
339.         # --- Run second model (solder) on the same image ---
340.         results_solder = self.yolo_model_solder(image_path, conf=0.95)
341.         # Overlay the second model's detections on top of the first
342.         annotated_img_solder = results_solder[0].plot()
343.         # Combine the two results
344.         # Simple way: blend the two images (alpha = 0.5 for the second)

```

```

345.         combined_img = cv2.addWeighted(annotated_img, 0.5, annotated_img_solder, 0.5, 0)
346.
347.         # --- Compute mask-based percentage ---
348.         if results_dev and results_solder: # ellenőrizzük, hogy vannak eredmények
349.             # Ellenőrizzük, hogy masks nem None
350.             if results_solder[0].masks is not None:
351.                 solder_mask = results_solder[0].masks.data.bool()
352.                 solder_pixels = solder_mask.sum().item()
353.             else:
354.                 solder_mask = None
355.                 solder_pixels = 0
356.
357.             if results_dev[0].masks is not None:
358.                 void_mask = results_dev[0].masks.data.bool()
359.                 # Csak a solder területén belüli void pixelek
360.                 if solder_mask is not None:
361.                     void_mask = void_mask & solder_mask
362.                 void_pixels = void_mask.sum().item()
363.             else:
364.                 void_pixels = 0
365.
366.             void_percent = (void_pixels / solder_pixels * 100) if solder_pixels > 0 else 0
367.             self.void_label.setText(f"Void: {void_percent:.2f}%")
368.             self.solder_label.setText(f"Solder: {100 - void_percent:.2f}%")
369.             print(f"Void a solder maskon belül: {void_percent:.2f}%")
370.             if void_percent > 10:
371.                 print("High void percentage detected!")
372.                 self.pass_label.setText("Fail")
373.                 self.pass_label.setStyleSheet("color: red; font-weight: bold; background-
color: transparent;")
374.
375.             else:
376.                 self.pass_label.setText("Pass")
377.                 self.pass_label.setStyleSheet("color: green; font-weight: bold; background-
color: transparent;")
378.
379.             else:
380.                 self.void_label.setText(f"Void: 0.00%")
381.                 self.solder_label.setText(f"Solder: 100.00%")
382.                 print(f"Void a solder maskon belül: 0.00%")
383.                 self.pass_label.setText("Pass")
384.                 self.pass_label.setStyleSheet("color: green; font-weight: bold; background-
color: transparent;")
385.
386.             # Convert to RGB for Qt
387.             rgb_img = cv2.cvtColor(combined_img, cv2.COLOR_BGR2RGB)
388.             h, w, ch = rgb_img.shape
389.             bytes_per_line = ch * w
390.             qt_img = QImage.fromImage(
391.                 QImage(rgb_img.data, w, h, bytes_per_line, QImage.Format_RGB888)
392.             )
393.
394.             # Update QLabel
395.             self.image_label.setPixmap(
396.                 qt_img.scaled(self.image_label.size(), Qt.KeepAspectRatio,
Qt.SmoothTransformation)
397.             )
398.
399.             # --- Optional: print stats ---
400.             num_detections_dev = len(results_dev[0].boxes)
401.             num_detections_solder = len(results_solder[0].boxes)
402.             print(f"Deviation detections: {num_detections_dev}")
403.             print(f"Solder detections: {num_detections_solder}")
404.
405.         def run_batch_analysis(self):
406.             if not hasattr(self, "image_files") or not self.image_files:

```

```

406.         print("Nincs betöltött mappa.")
407.         return
408.
409.         # Gombok inaktiválása
410.         self.auto_btn.setEnabled(False)
411.         self.save_results_btn.setEnabled(False)
412.         self.prev_btn.setEnabled(False)
413.         self.next_btn.setEnabled(False)
414.
415.
416.         if self.thumbnail.layout() is not None:
417.             old_layout = self.thumbnail.layout()
418.             while old_layout.count():
419.                 item = old_layout.takeAt(0)
420.                 widget = item.widget()
421.                 if widget:
422.                     widget.setParent(None)
423.             QWidget().setLayout(old_layout)
424.
425.         # Új progress bar létrehozása
426.         progress_bar = QProgressBar()
427.         progress_bar.setRange(0, len(self.image_files))
428.         progress_bar.setValue(0)
429.         progress_bar.setTextVisible(True)
430.         progress_bar.setFixedHeight(40) # Magasság
431.         progress_bar.setFormat("Processing: %p")
432.         progress_bar.setStyleSheet("""
433.             QProgressBar {
434.                 border: 2px solid #555;
435.                 border-radius: 10px;
436.                 background-color: #DDD;
437.                 text-align: center;
438.                 font-weight: bold;
439.                 color: black;
440.             }
441.             QProgressBar::chunk {
442.                 border-radius: 10px;
443.                 background-color: qlineargradient(
444.                     x1:0, y1:0, x2:1, y2:0,
445.                     stop:0 #05CC08, stop:1 #05CC1C
446.                 );
447.                 margin: 1px;
448.             }
449.         """)
450.
451.         # Új layout és hozzáadás
452.         progress_layout = QVBoxLayout()
453.         progress_layout.addWidget(progress_bar)
454.         self.thumbnail.setLayout(progress_layout)
455.
456.         self.analysis_results = [] # reseteljük, új futás
457.
458.
459.         folder_path = os.path.dirname(self.image_files[0])
460.         output_folder = os.path.join(folder_path, "results")
461.         os.makedirs(output_folder, exist_ok=True)
462.
463.         for idx, img_path in enumerate(self.image_files):
464.             print(f"Feldolgozás: {img_path} ({idx+1}/{len(self.image_files)})")
465.
466.             self.current_image_index = idx
467.
468.
469.             filename = os.path.splitext(os.path.basename(img_path))[0]
470.             filename = filename.split("_")[0] if "_" in filename else filename

```

```

471.         self.position_label.setText(f"Position: {filename}")
472.
473.         # Plane section
474.         folder_path = os.path.dirname(img_path)
475.         folder_path_name = os.path.basename(folder_path)
476.         if folder_path_name.startswith("IC"):
477.             plane_name = folder_path_name.split("_")[-1]
478.             self.plane_section_label.setText(f"Plane section: {plane_name}")
479.         else:
480.             self.plane_section_label.setText(f"Plane section: {folder_path_name}")
481.
482.         # IC type
483.         parent_folder_path = os.path.dirname(os.path.dirname(img_path))
484.         parent_folder = os.path.basename(parent_folder_path)
485.         self.IC_type_label.setText(f"IC type: {parent_folder}")
486.
487.         self.run_analysis() # meglévő logika kiszámolja a void/solder százalékokat,
pass/fail-t
488.
489.         progress_bar.setValue(idx + 1)
490.         QApplication.processEvents() # GUI frissítése
491.
492.
493.         image_name = os.path.basename(img_path)
494.         plane_section = self.plane_section_label.text().replace("Plane section: ", "")
495.         ic_type = self.IC_type_label.text().replace("IC type: ", "")
496.         position = self.position_label.text().replace("Position: ", "")
497.         status = self.pass_label.text()
498.         solder_percent = self.solder_label.text().replace("Solder: ", "").replace("%",
""")
499.         void_percent = self.void_label.text().replace("Void: ", "").replace("%", "")
500.
501.
502.         annotated_img_path = os.path.join(output_folder, f"annotated_{image_name}")
503.         pixmap = self.image_label.pixmap()
504.         if pixmap:
505.             pixmap.save(annotated_img_path)
506.
507.         entry = {
508.             "image_name": image_name,
509.             "plane_section": plane_section,
510.             "ic_type": ic_type,
511.             "position": position,
512.             "solder_percent": float(solder_percent),
513.             "void_percent": float(void_percent),
514.             "status": status,
515.             "annotated_image": annotated_img_path
516.         }
517.
518.         self.analysis_results.append(entry)
519.
520.         # JSON fájl írása
521.         json_path = os.path.join(output_folder, "analysis_results.json")
522.         with open(json_path, "w", encoding="utf-8") as f:
523.             import json
524.             json.dump(self.analysis_results, f, ensure_ascii=False, indent=4)
525.
526.
527.         # Gombok újra aktívak
528.         self.auto_btn.setEnabled(True)
529.         self.save_results_btn.setEnabled(True)
530.         self.prev_btn.setEnabled(True)
531.         self.next_btn.setEnabled(True)
532.
533.

```

```

534.         old_layout = self.thumbnail.layout()
535.         if old_layout is not None:
536.             while old_layout.count():
537.                 item = old_layout.takeAt(0)
538.                 widget = item.widget()
539.                 if widget:
540.                     widget.setParent(None)
541.
542.         # Thumbnail visszaállítása alapállapotba
543.         self.thumbnail.setText("")
544.         self.thumbnail.setStyleSheet("background-color: transparent; border: 0px solid #666;
font-family: 'Cooper Black'; font-size: 12px;")
545.         self.thumbnail.setAlignment(Qt.AlignCenter)
546.
547.         print(f"Eredmények elmentve: {json_path}")
548.
549.
550.
551.     def export_pdf_from_json(self, json_path, output_pdf, operator_name=None):
552.         with open(json_path, "r", encoding="utf-8") as f:
553.             data = json.load(f)
554.
555.             doc = SimpleDocTemplate(output_pdf, pagesize=A4)
556.             styles = getSampleStyleSheet()
557.             story = []
558.
559.             # --- Bevezető oldal ---
560.             # Címsor
561.             intro_title = "<b>Soldering Inspector</b>"
562.             story.append(Spacer(1, 4*cm))
563.             story.append(Paragraph(intro_title, styles['Title']))
564.             story.append(Spacer(1, 1*cm))
565.
566.             # Ikon kép
567.             icon_path = "assets/Icon.png"
568.             if os.path.exists(icon_path):
569.                 story.append(Image(icon_path, width=200, height=200))
570.                 story.append(Spacer(1, 1*cm))
571.
572.             # Operátor neve
573.             if operator_name:
574.                 story.append(Paragraph(f"<b>Operator:</b> {operator_name}", styles['Normal']))
575.                 story.append(Spacer(1, 2*cm))
576.
577.             story.append(PageBreak())
578.
579.             # --- Eredmények oldalai ---
580.             for idx, entry in enumerate(data):
581.                 status = entry.get("status", "N/A")
582.                 status_color = "green" if status.lower() == "pass" else "red" if status.lower()
== "fail" else "black"
583.
584.                 info_table_data = [
585.                     ["IC Type:", entry.get("ic_type", "N/A")],
586.                     ["Plane Section:", entry.get("plane_section", "N/A")],
587.                     ["Position:", entry.get("position", "N/A")],
588.                     ["Solder %:", f"{entry.get('solder_percent', 0):.2f}%"],
589.                     ["Void %:", f"{entry.get('void_percent', 0):.2f}%"],
590.                     ["Status:", Paragraph(f"<font
color='{status_color}'><b>{status}</b></font>", styles['Normal'])]
591.                 ]
592.                 info_table = Table(info_table_data, colWidths=[100, 300])
593.                 info_table.setStyle(TableStyle([
594.                     ('BACKGROUND', (0, 0), (0, -1), colors.lightgrey),

```

```

595.         ('TEXTCOLOR', (0, 0), (-1, -1), colors.black),
596.         ('INNERGRID', (0, 0), (-1, -1), 0.25, colors.grey),
597.         ('BOX', (0, 0), (-1, -1), 0.25, colors.black)
598.     ]))
599.
600.     story.append(info_table)
601.     story.append(Spacer(1, 12))
602.
603.     img_path = entry.get("annotated_image", None)
604.     if img_path and os.path.exists(img_path):
605.         try:
606.             img = Image(img_path, width=400, height=400)
607.             story.append(img)
608.             story.append(Spacer(1, 24))
609.         except Exception as e:
610.             story.append(Paragraph(f"<i>Image load failed: {e}</i>",
styles['Normal']))
611.             story.append(Spacer(1, 12))
612.
613.         # Oldaltörés minden entry után, kivéve az utolsót
614.         if idx < len(data) - 1:
615.             story.append(PageBreak())
616.
617.         # --- Oldalszám hozzáadása ---
618.         def add_page_number(canvas, doc):
619.             page_num = canvas.getPageNumber()
620.             text = f"{page_num}"
621.             canvas.setFont('Helvetica', 10)
622.             canvas.drawRightString(A4[0]-2*cm, 1*cm, text)
623.
624.         doc.build(story, onFirstPage=add_page_number, onLaterPages=add_page_number)
625.         print(f"PDF exportálva ide: {output_pdf}")
626.
627.     def on_save_results_clicked(self):
628.
629.         if not hasattr(self, "image_files") or not self.image_files:
630.             print("Nincs feldolgozott mappa.")
631.             return
632.
633.         # Találja meg a results mappát
634.         folder_path = os.path.dirname(self.image_files[0])
635.         output_folder = os.path.join(folder_path, "results")
636.         json_path = os.path.join(output_folder, "analysis_results.json")
637.
638.         if not os.path.exists(json_path):
639.             print(f"JSON fájl nem található: {json_path}")
640.             return
641.
642.         # Bekéri az operátor nevét
643.         # Létrehozunk egy custom QDialogDialog-ot
644.         dialog = QDialogDialog(self)
645.         dialog.setWindowTitle("Operator Name")
646.         dialog.setLabelText("Enter operator name:")
647.         dialog.setTextValue("") # alapértelmezett érték
648.         dialog.setOkButtonText("Submit")
649.         dialog.setCancelButtonText("Cancel")
650.
651.
652.         dialog.setStyleSheet("""
653.             QDialog {
654.                 background-color: #12121f;
655.                 color: #ffffff;
656.                 font-family: 'Segoe UI';
657.                 font-size: 18px;
658.             }

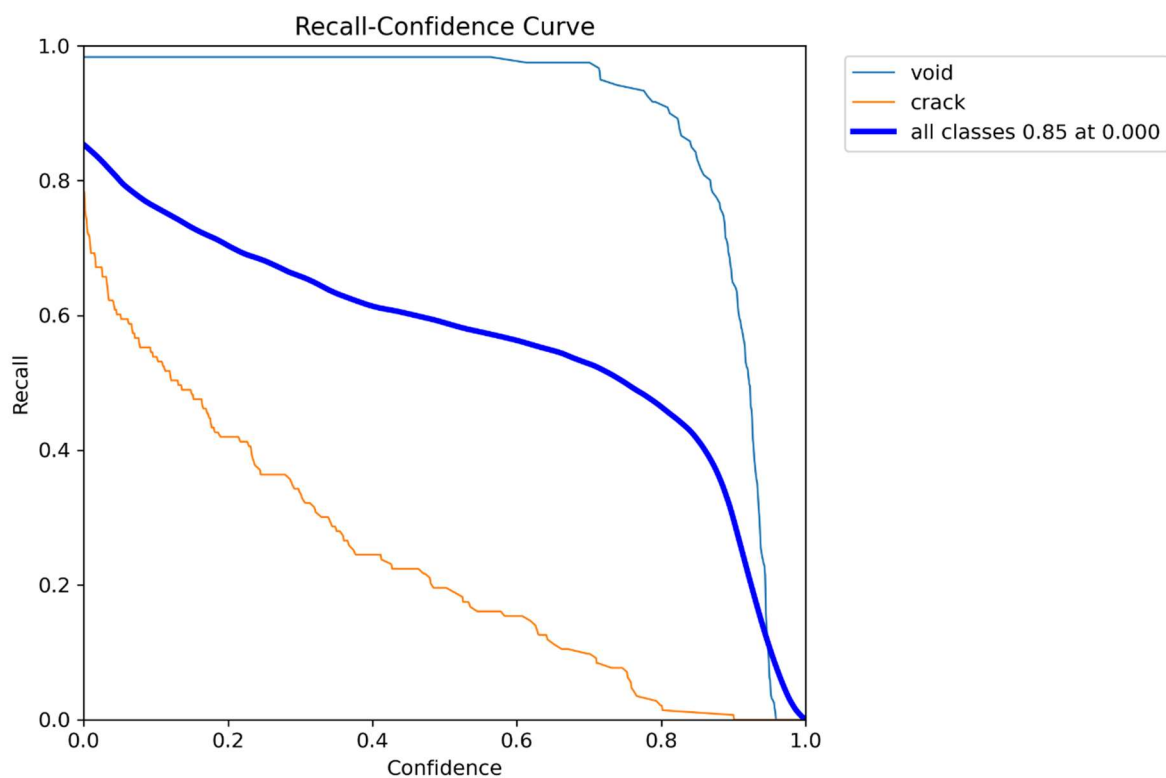
```

```

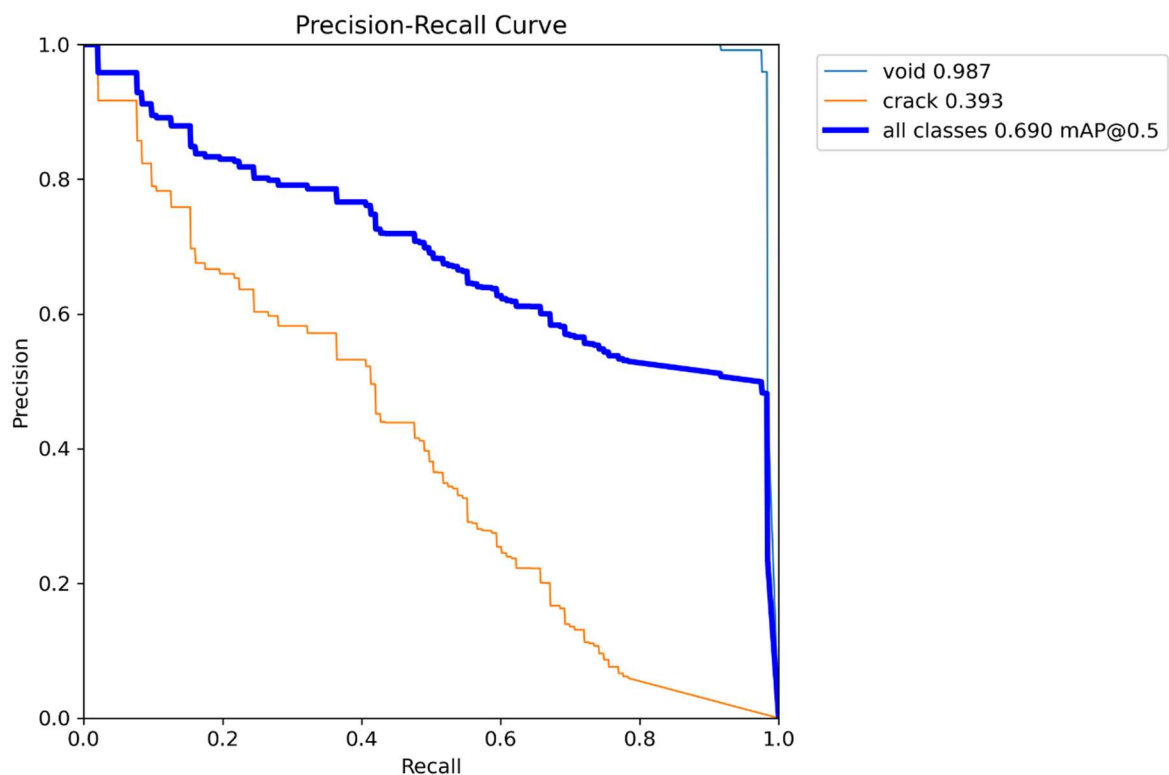
659.         QLabel {
660.             color: #ffffff;
661.             font-size: 20px;
662.         }
663.         QLineEdit {
664.             background-color: #1e1e2f;
665.             color: #ffffff;
666.             border: 2px solid #4f46e5;
667.             border-radius: 10px;
668.             padding: 8px;
669.             font-size: 18px;
670.         }
671.         QPushButton {
672.             background-color: #1e1e2f;
673.             color: #ffffff;
674.             border-radius: 12px;
675.             padding: 8px 15px;
676.             font-weight: bold;
677.             font-size: 16px;
678.         }
679.         QPushButton:hover {
680.             background-color: #4f46e5;
681.         }
682.         QPushButton:pressed {
683.             background-color: #37364f;
684.         }
685.     """)
686.
687.     operator_name, ok = QInputDialog.getText(dialog, "Operator Name", "Enter operator
name:")
688.     if not ok or not operator_name.strip():
689.         return
690.
691.     # Mentési hely kiválasztása
692.     save_path, _ = QFileDialog.getSaveFileName(
693.         self,
694.         "Save PDF",
695.         "",
696.         "PDF Files (*.pdf)"
697.     )
698.     if not save_path:
699.         return
700.
701.     # PDF generálása
702.     try:
703.         self.export_pdf_from_json(json_path, save_path,
operator_name=operator_name.strip())
704.         print(f"PDF successfully saved: {save_path}")
705.     except Exception as e:
706.         print(f"Failed to export PDF: {e}")
707.
708.

```

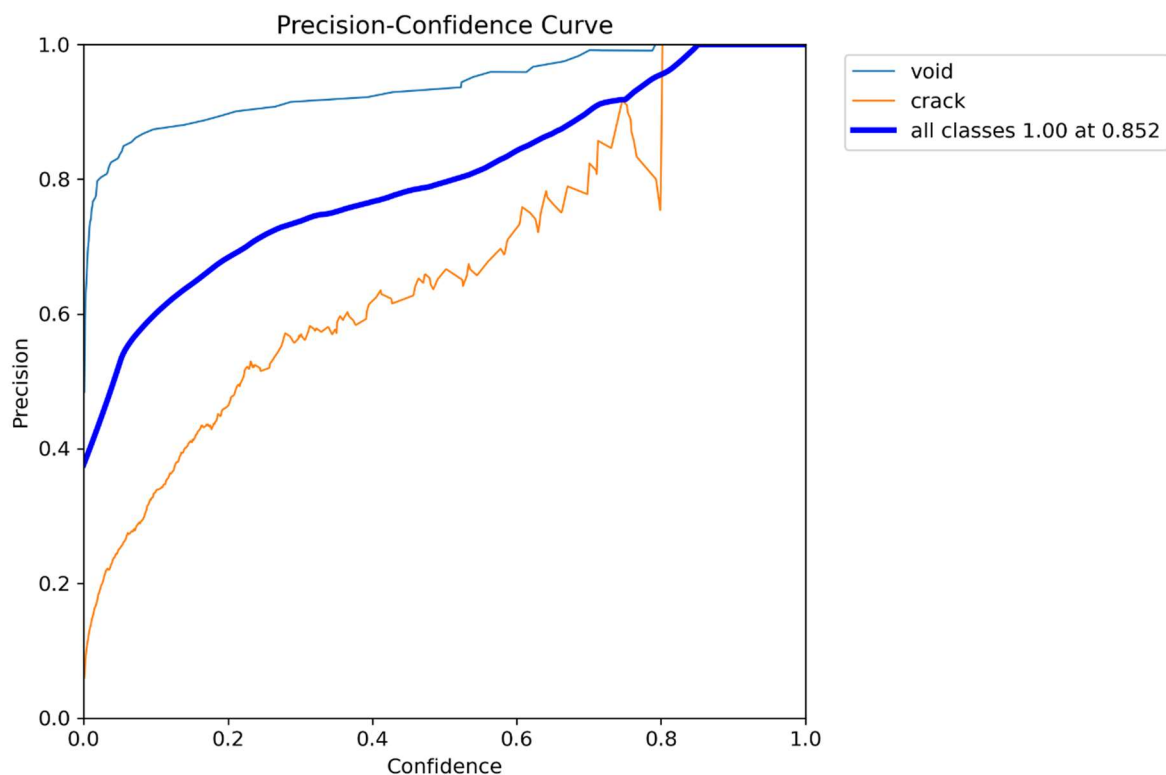
9.7 Az első tanításhoz tartozó képek és diagrammok



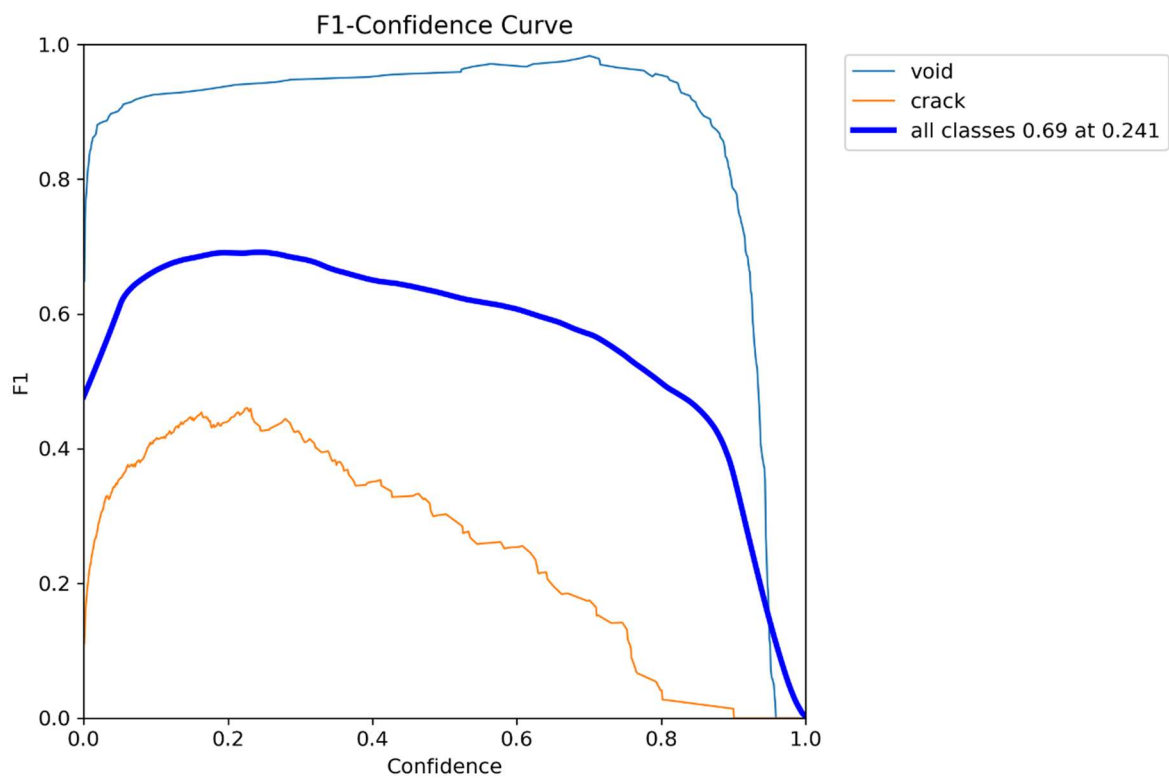
22. ábra Recall - Confidence diagram



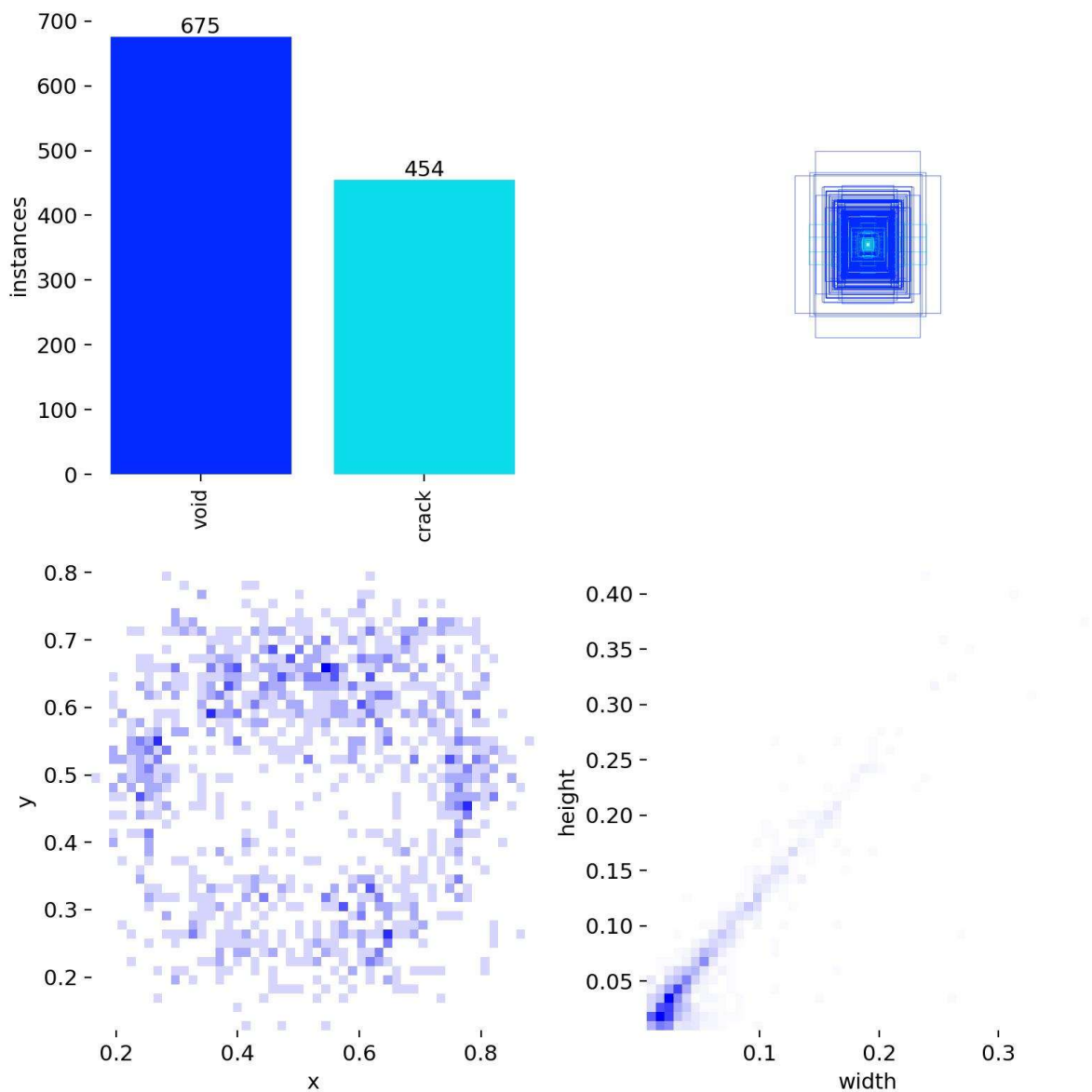
23. ábra Precision - Recall diagram



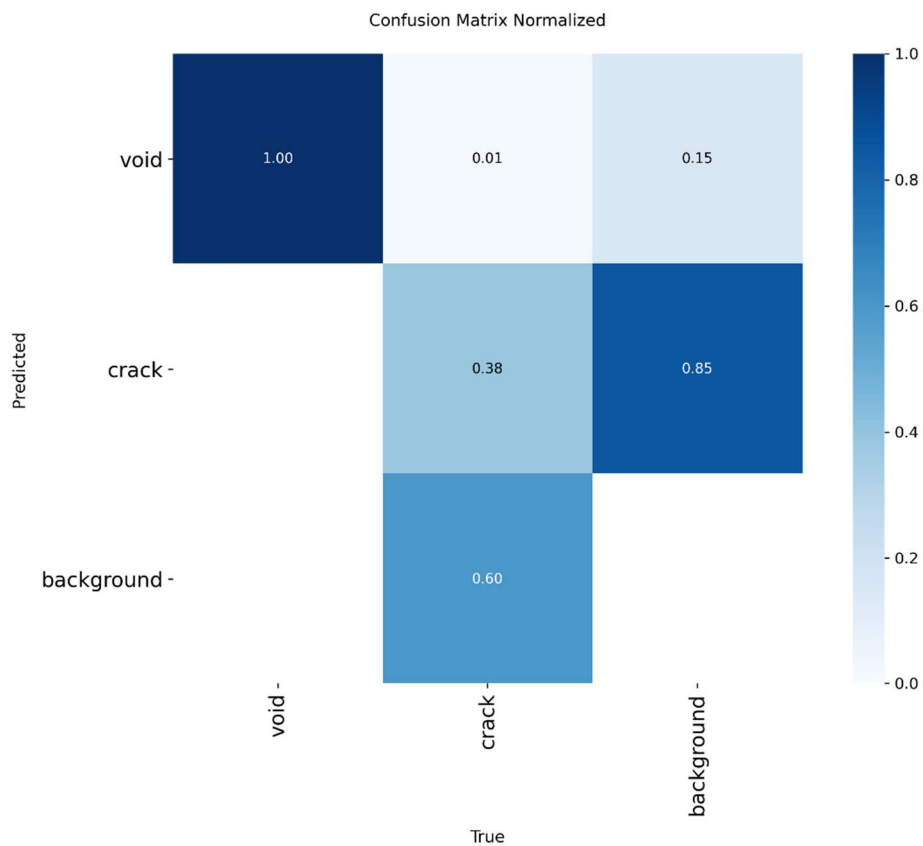
24. ábra Precision - Confidence diagram



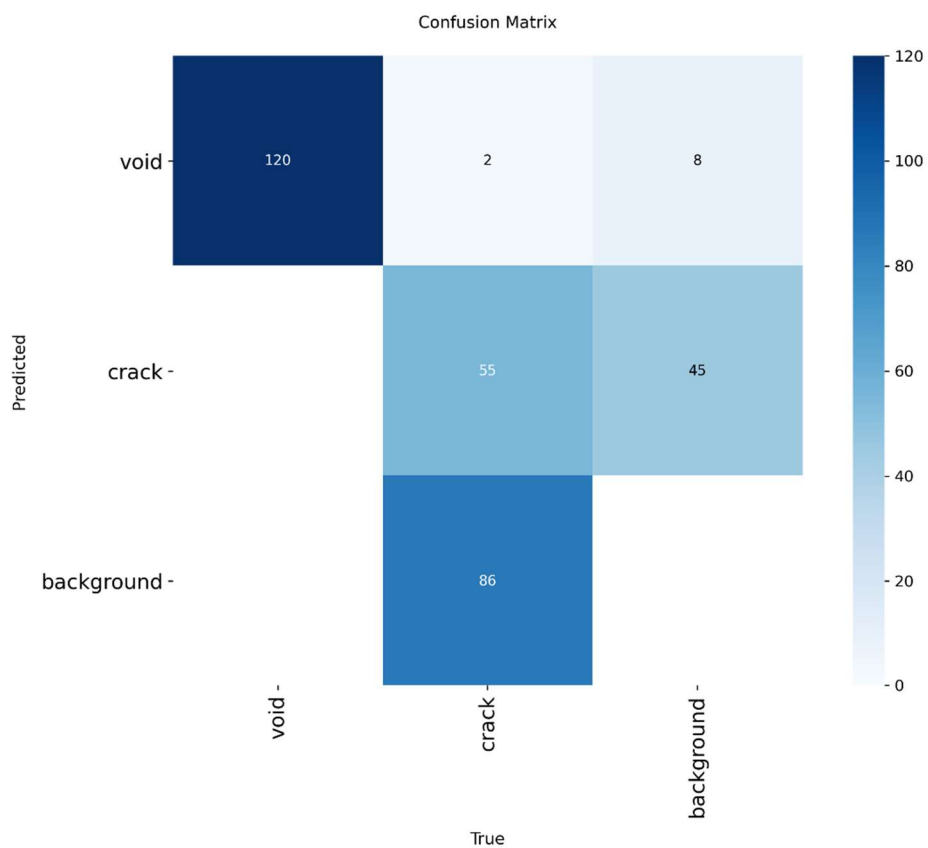
25. ábra F1 - Confidence diagram



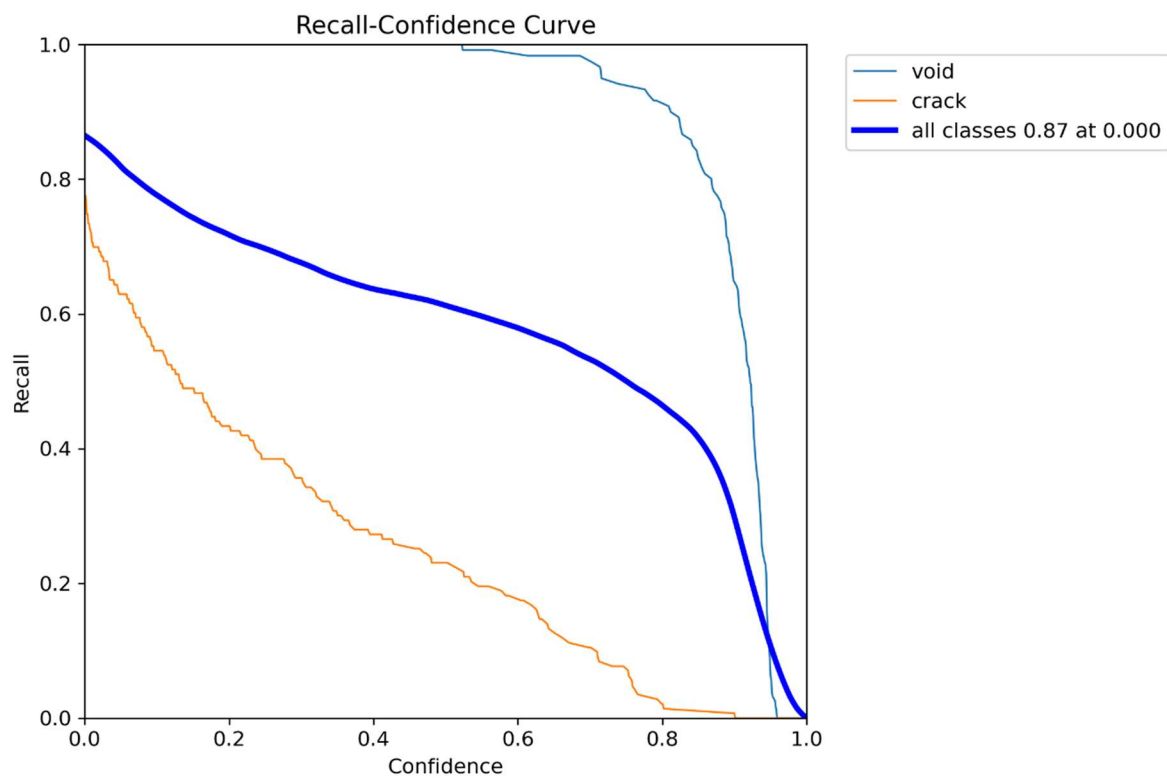
26. ábra Címkézési statisztika diagram



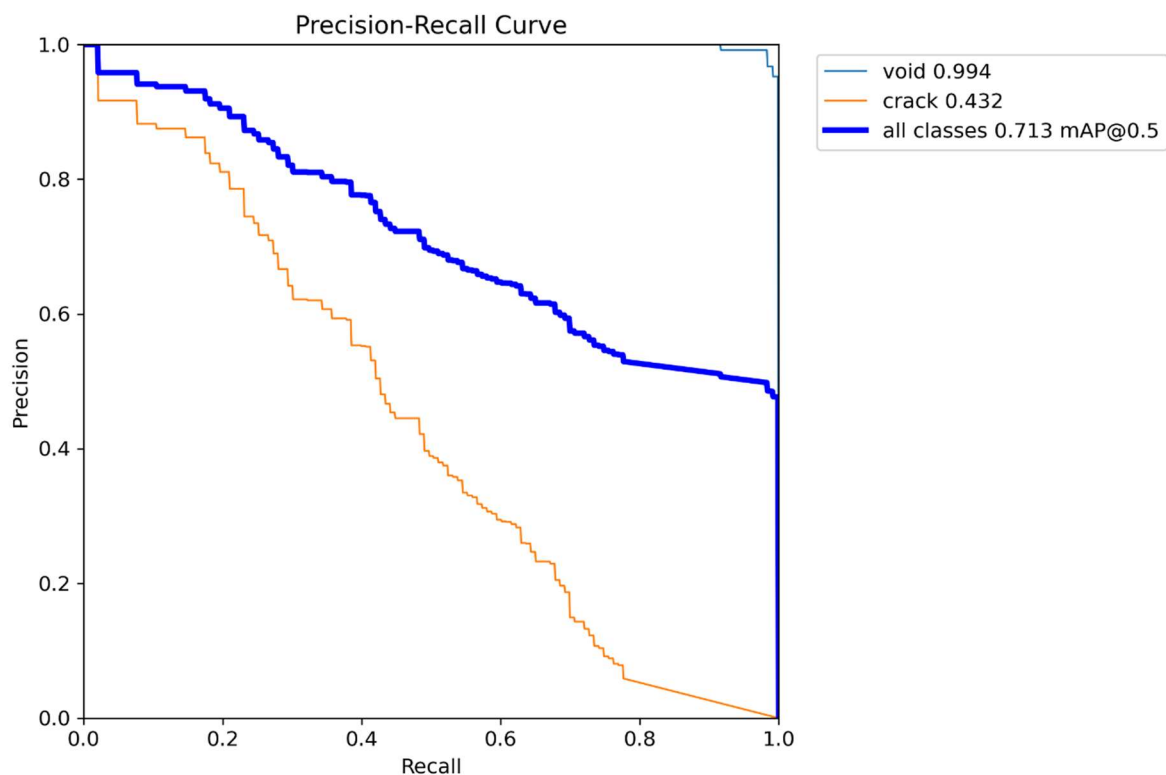
27. ábra *Confusion Matrix Normalized*



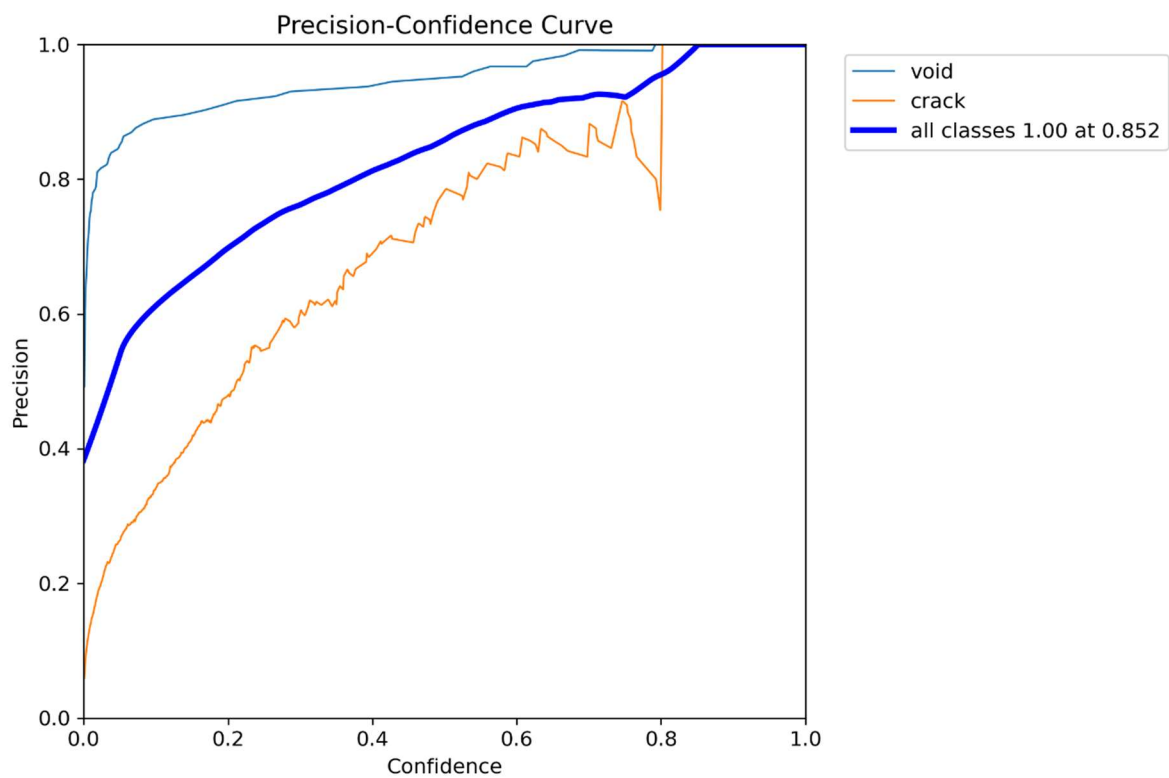
28. ábra Confusion Matrix



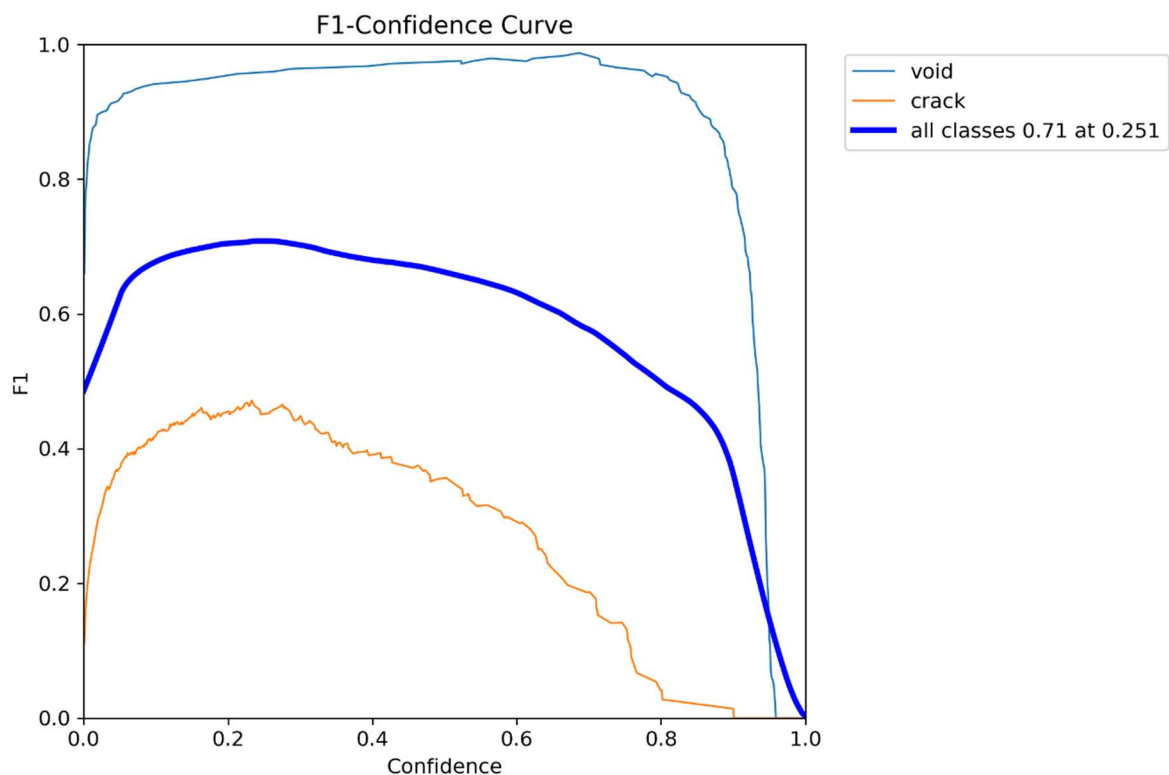
29. ábra Mask Recall - Confidence diagram



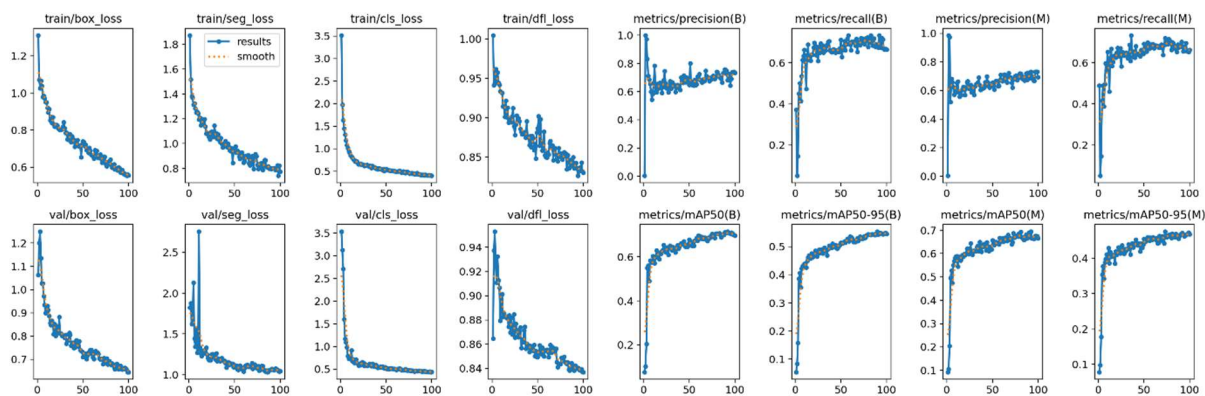
30. ábra Mask Precision - Recall diagram



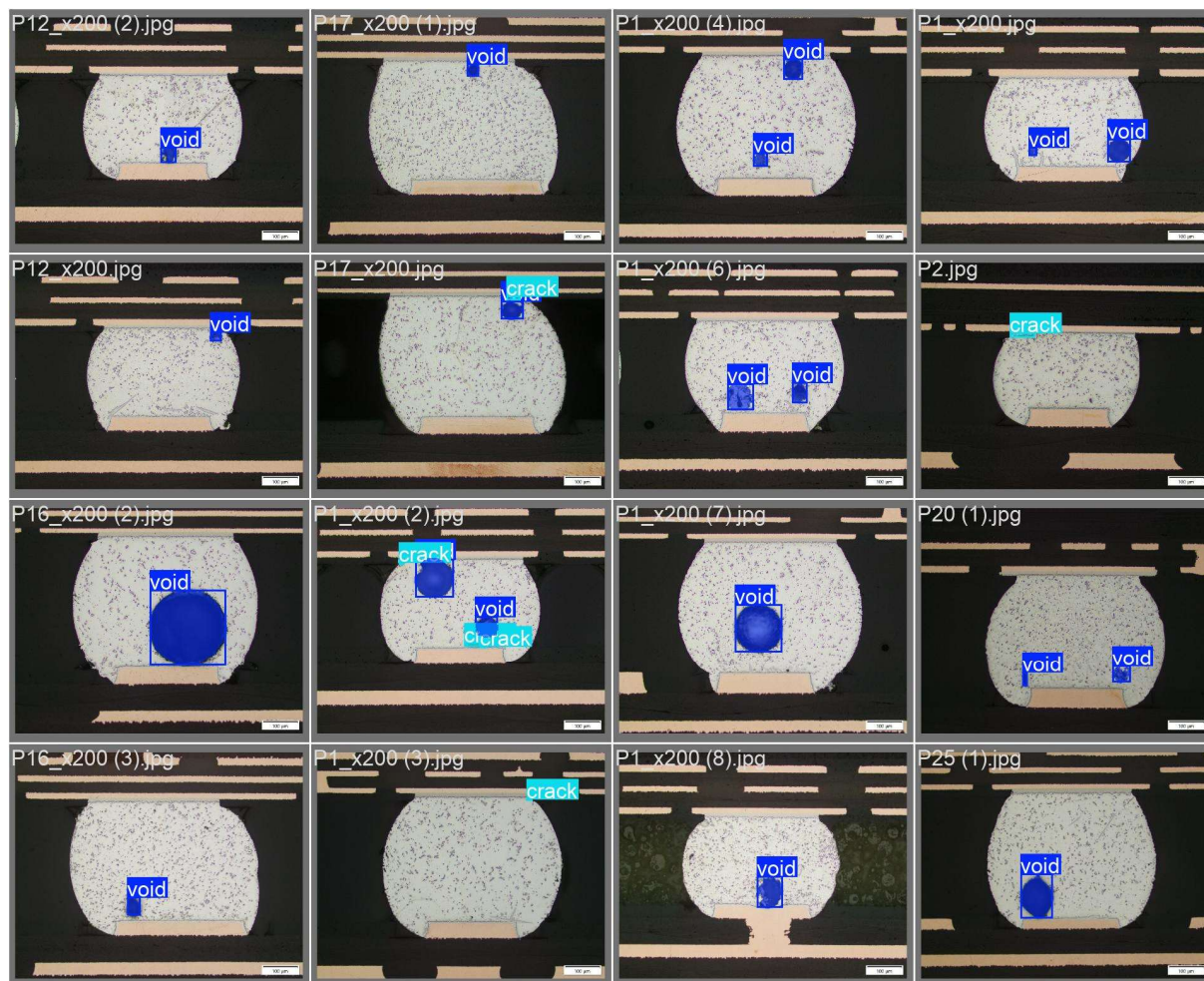
31. ábra Mask Precision - Confidence diagram



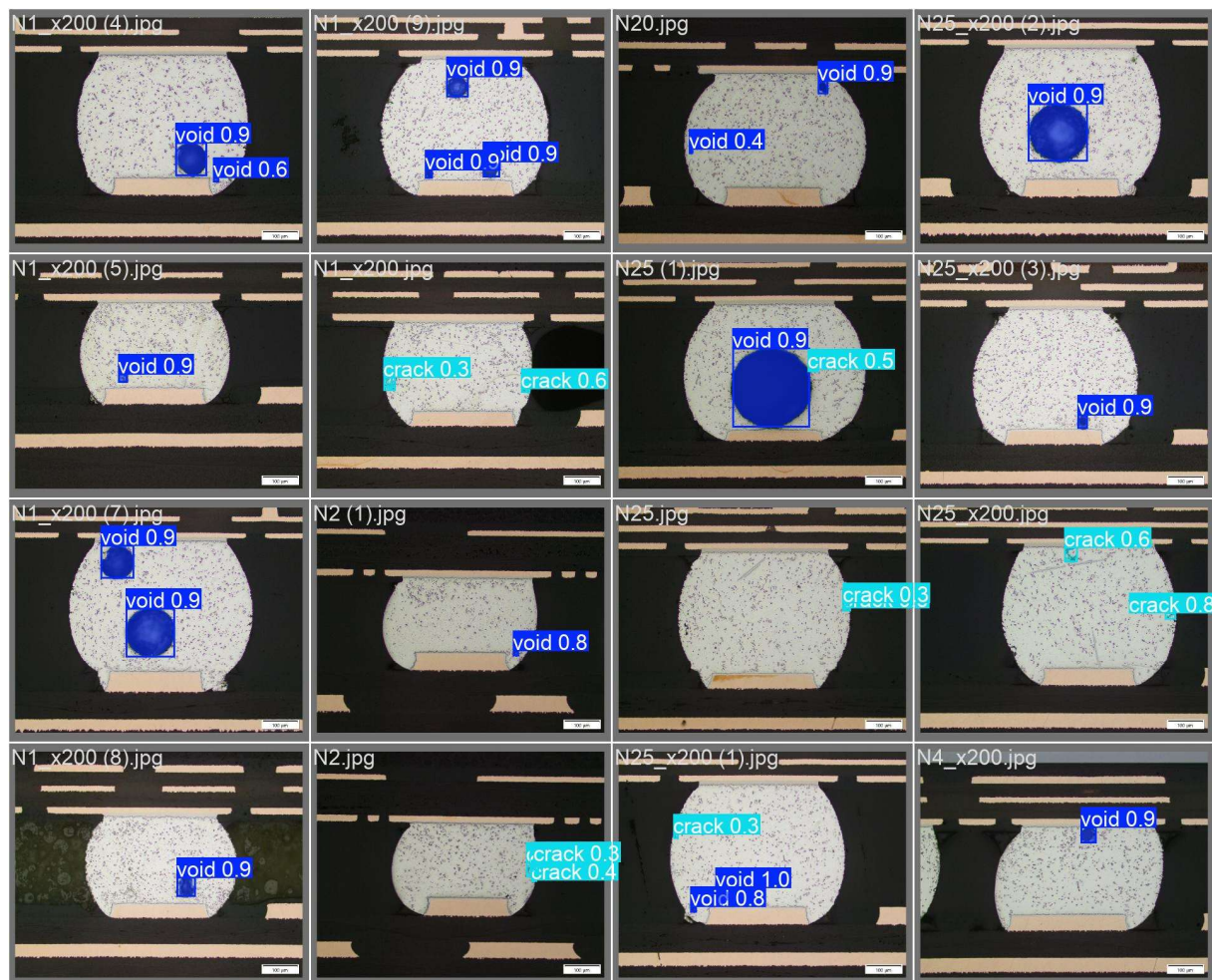
32. ábra Mask F1 - Confidence diagram



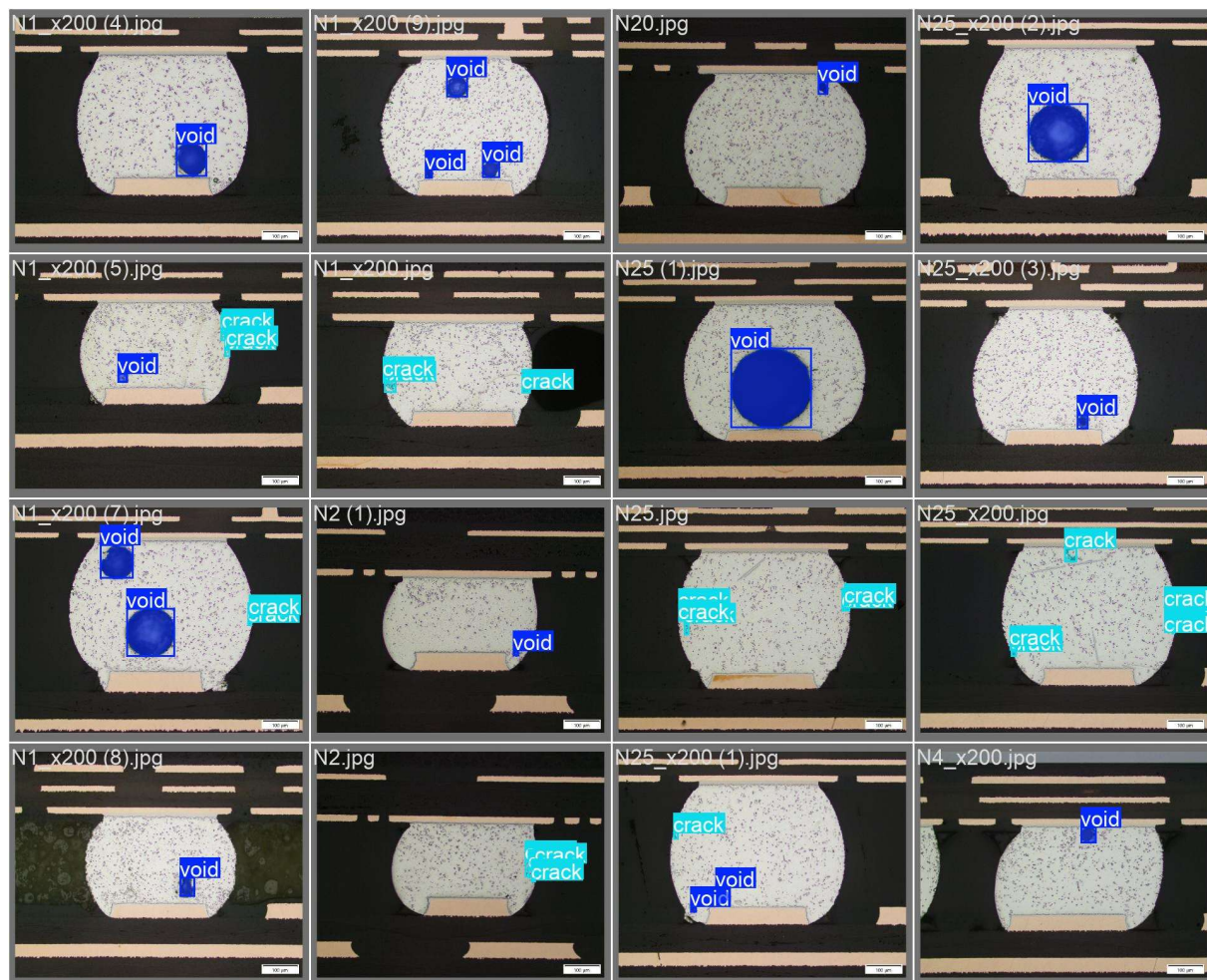
33. ábra Veszteség eredmények



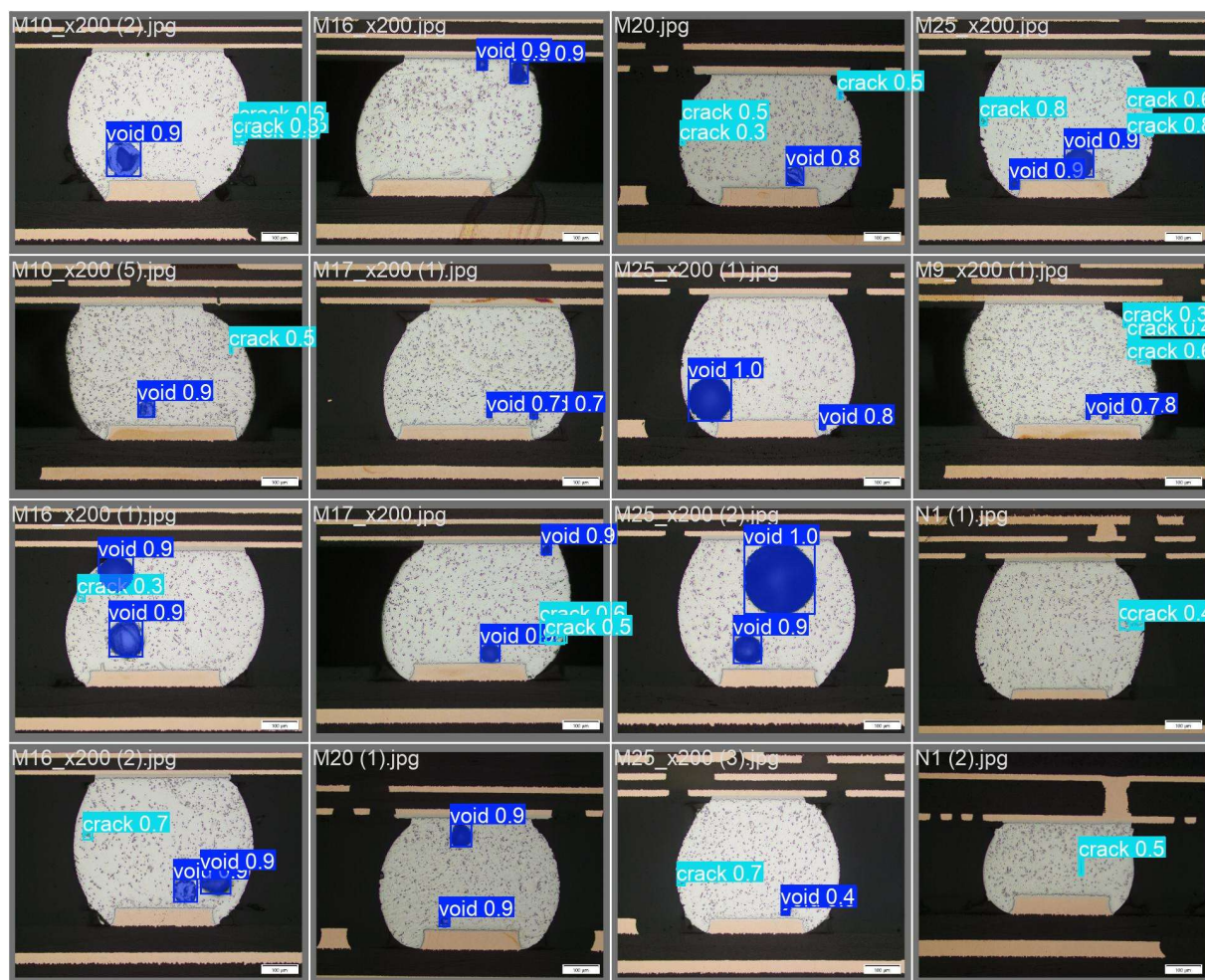
34. ábra Kézzel címkézett képek -2. köteg



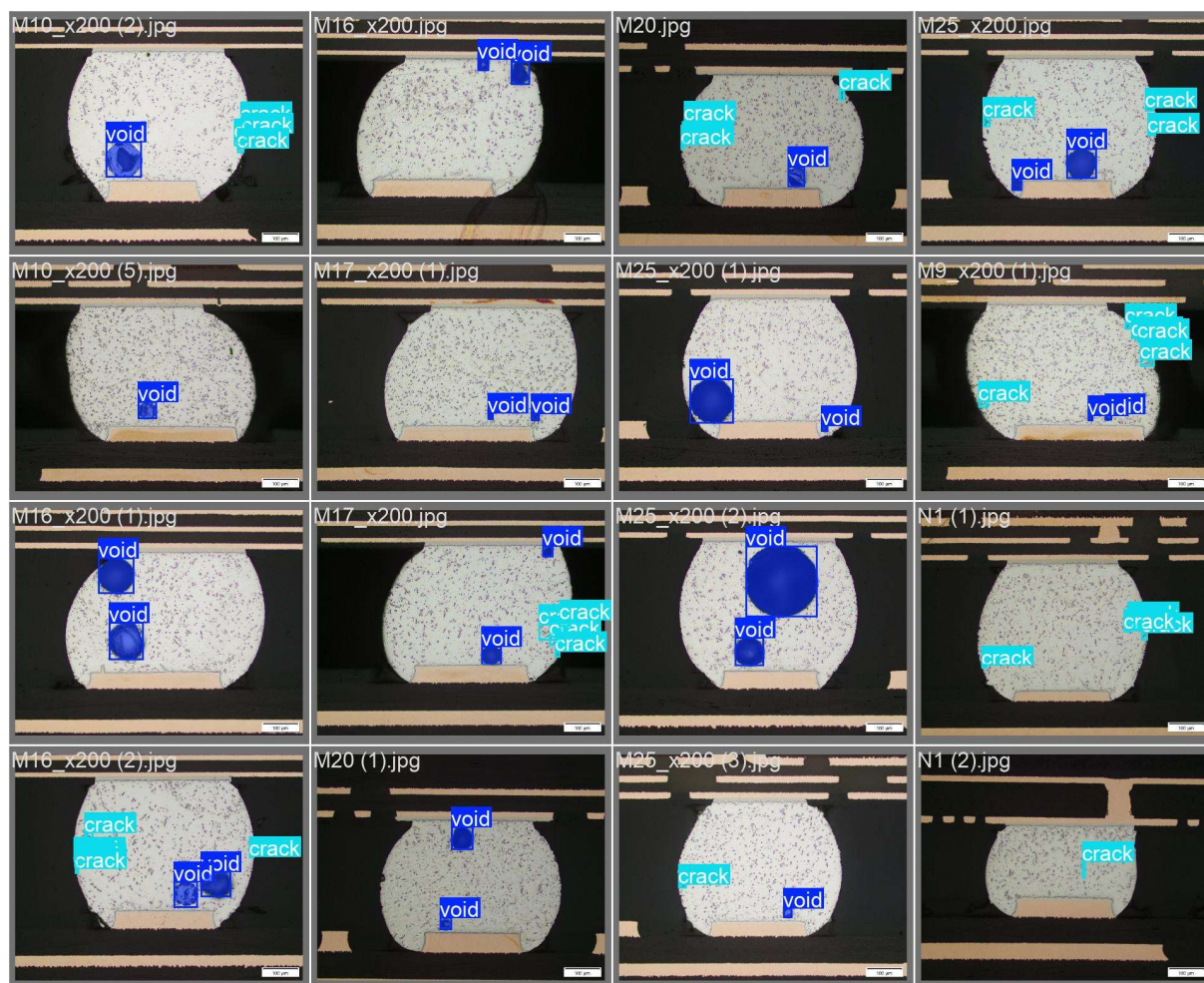
35. ábra Prediktált képek - 1. köteg



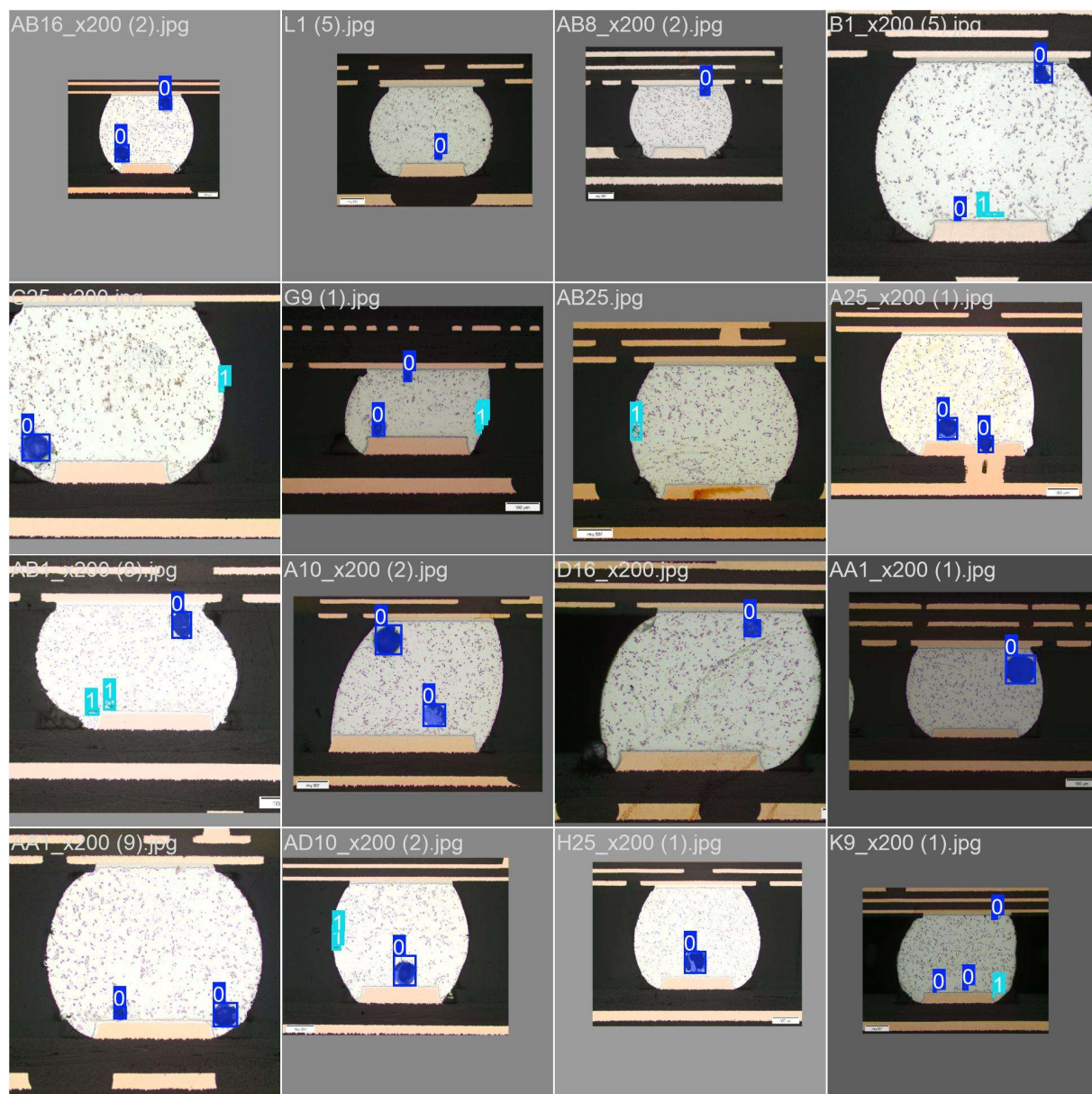
36. ábra Kézzel címkézett képek - 1. köteg



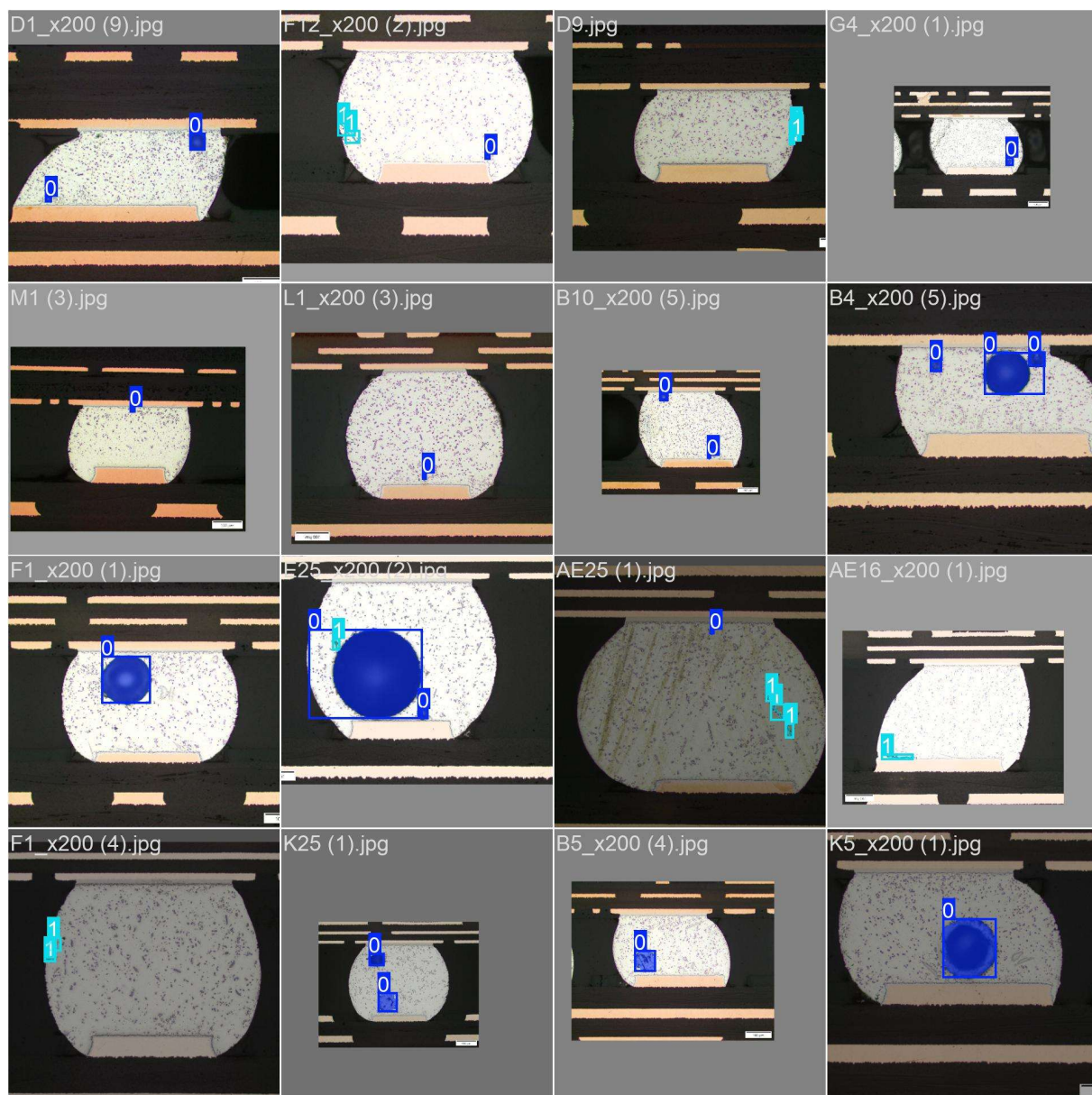
37. ábra Prediktált képek -0. köteg



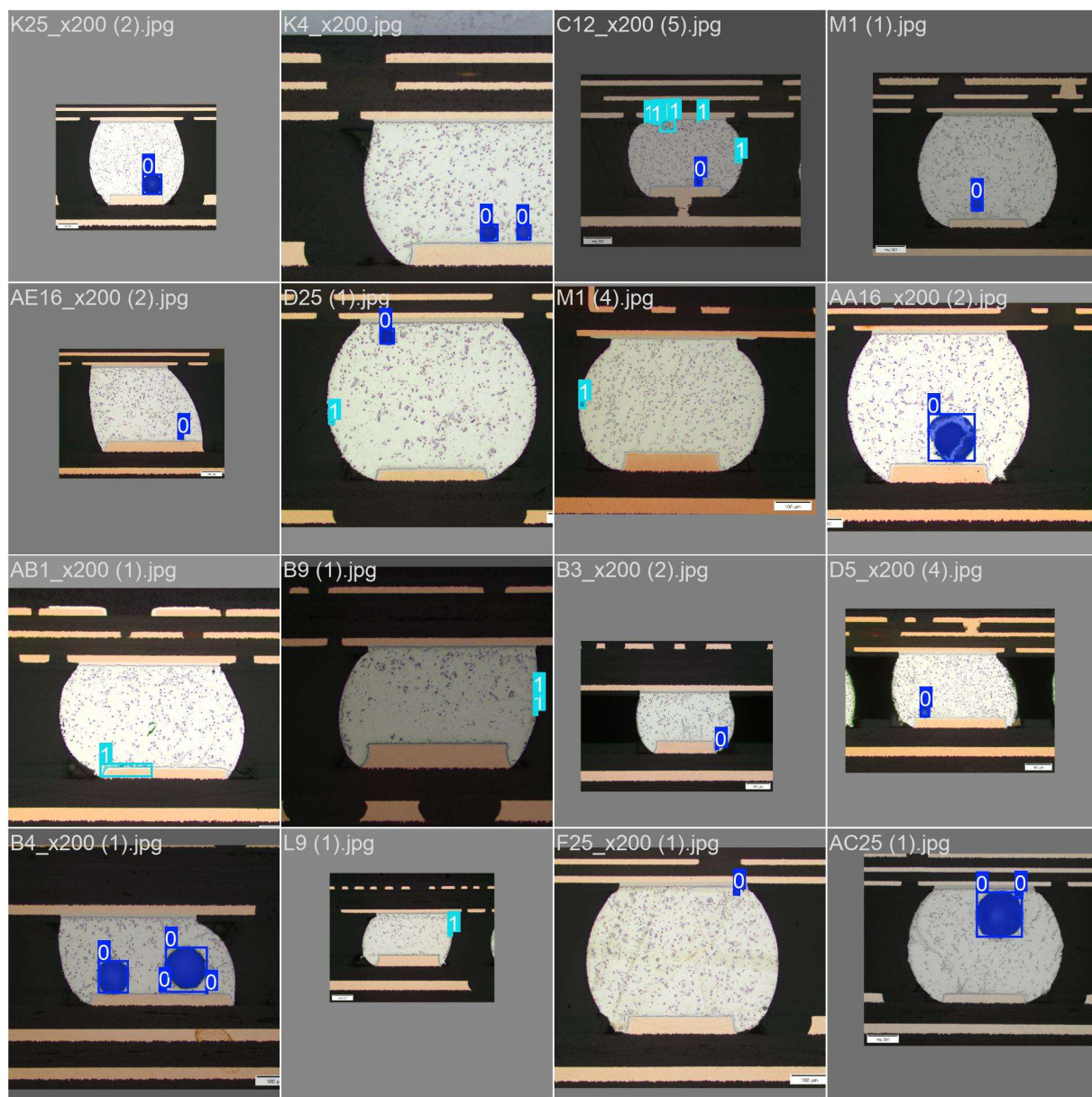
38. ábra Kézzel címkézett képek -0. köteg



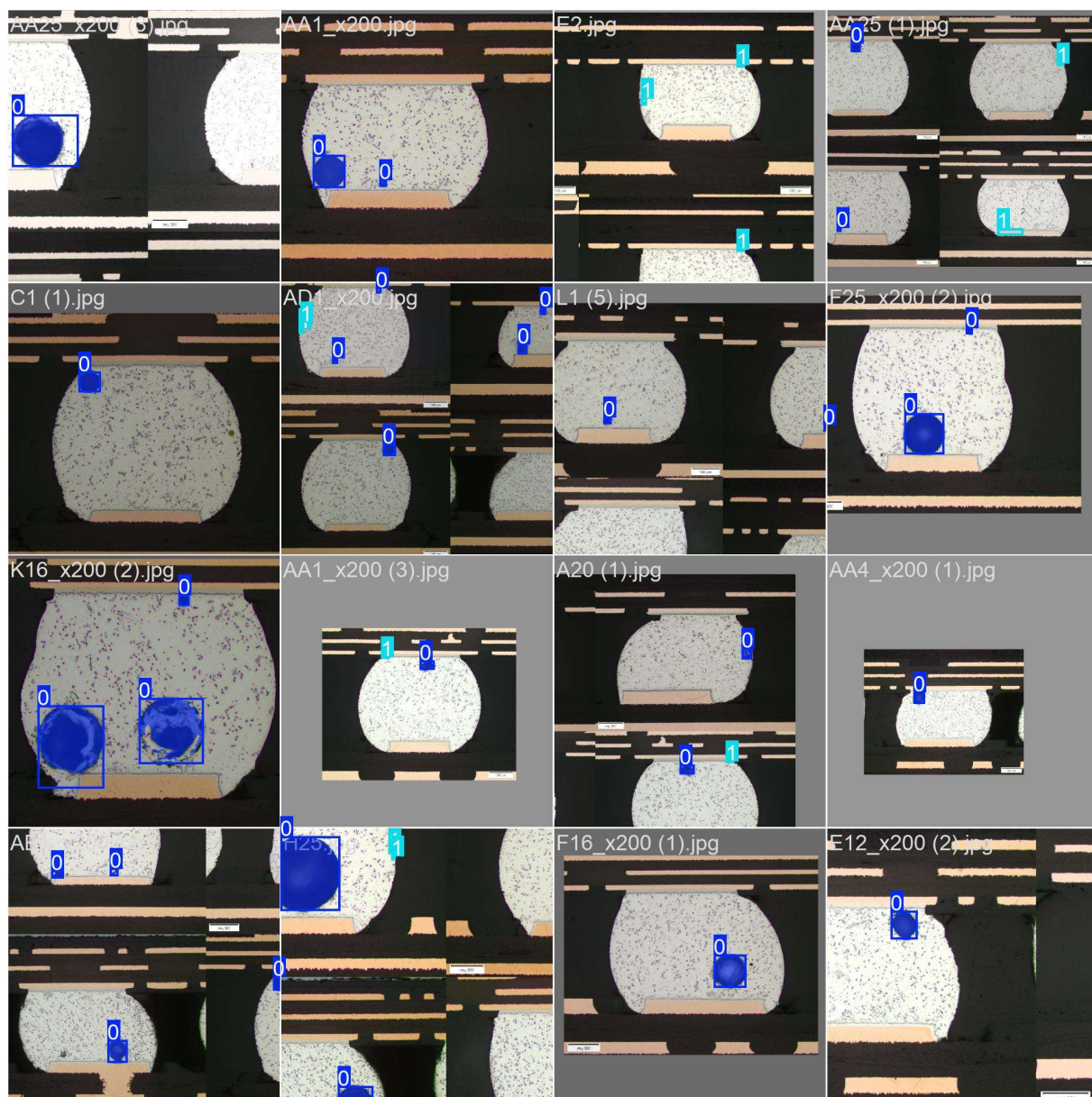
39. ábra Tréning köteg 2



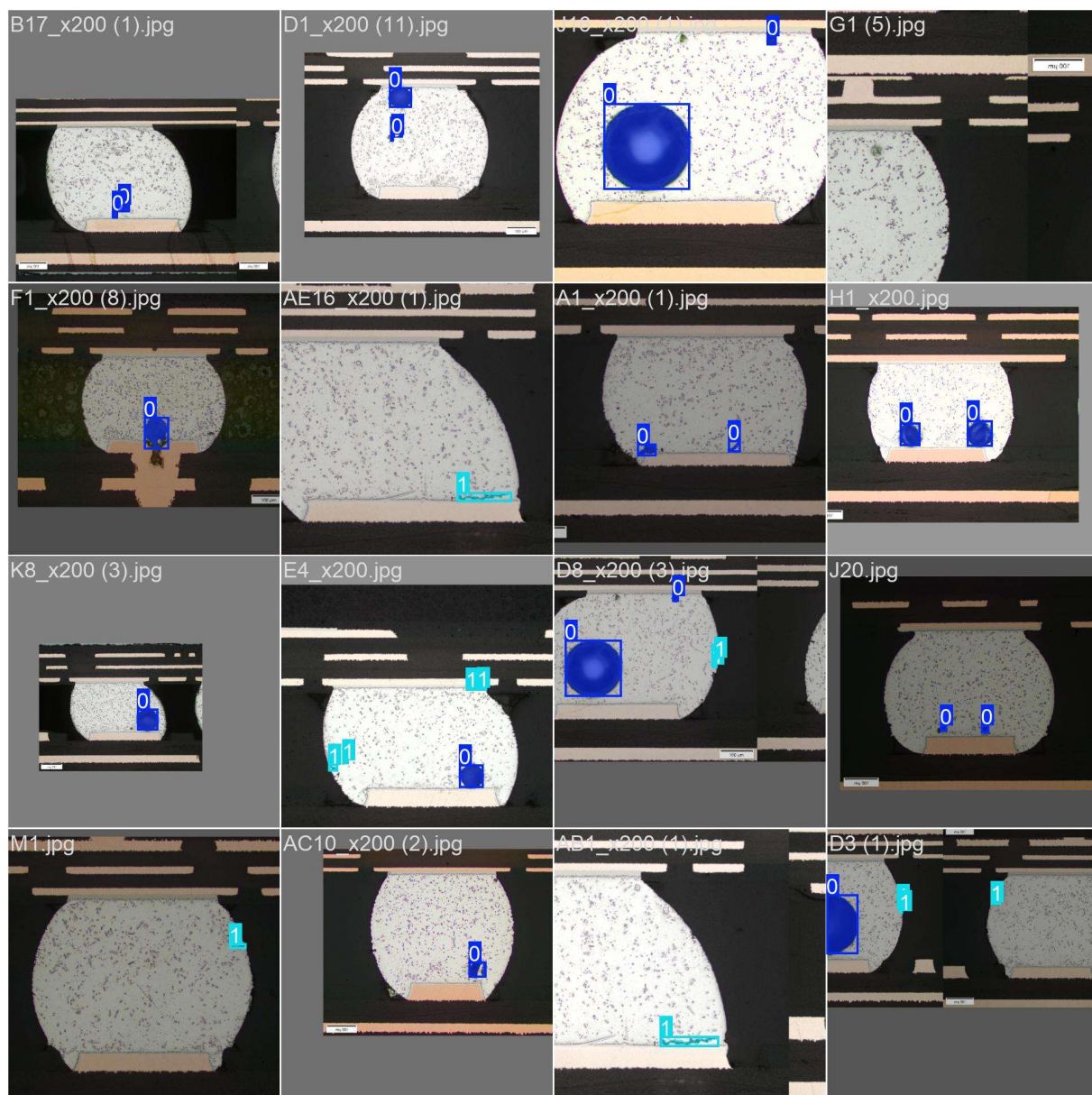
40. ábra Tréning köteg 1



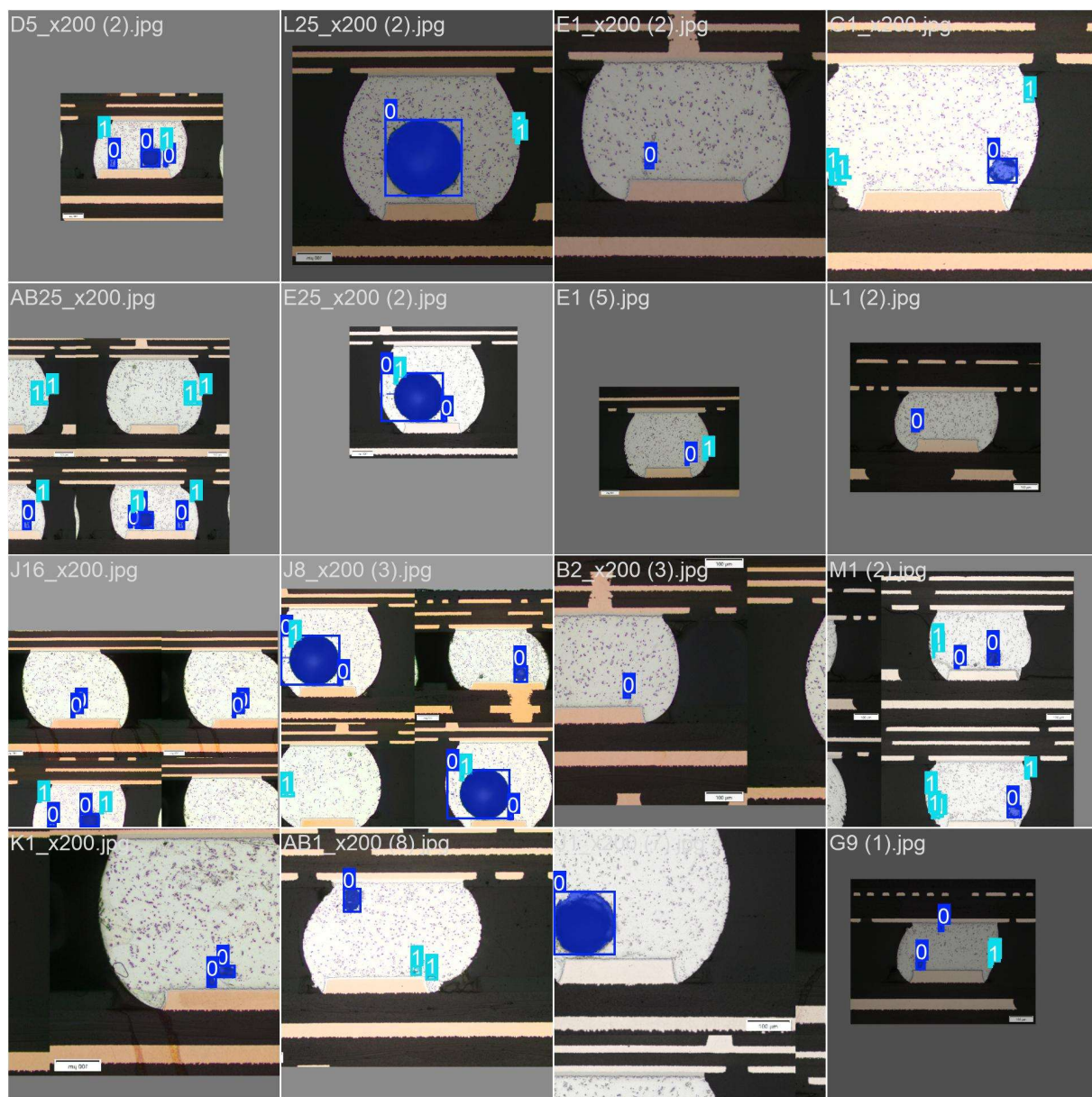
41. ábra Tréning köteg 0



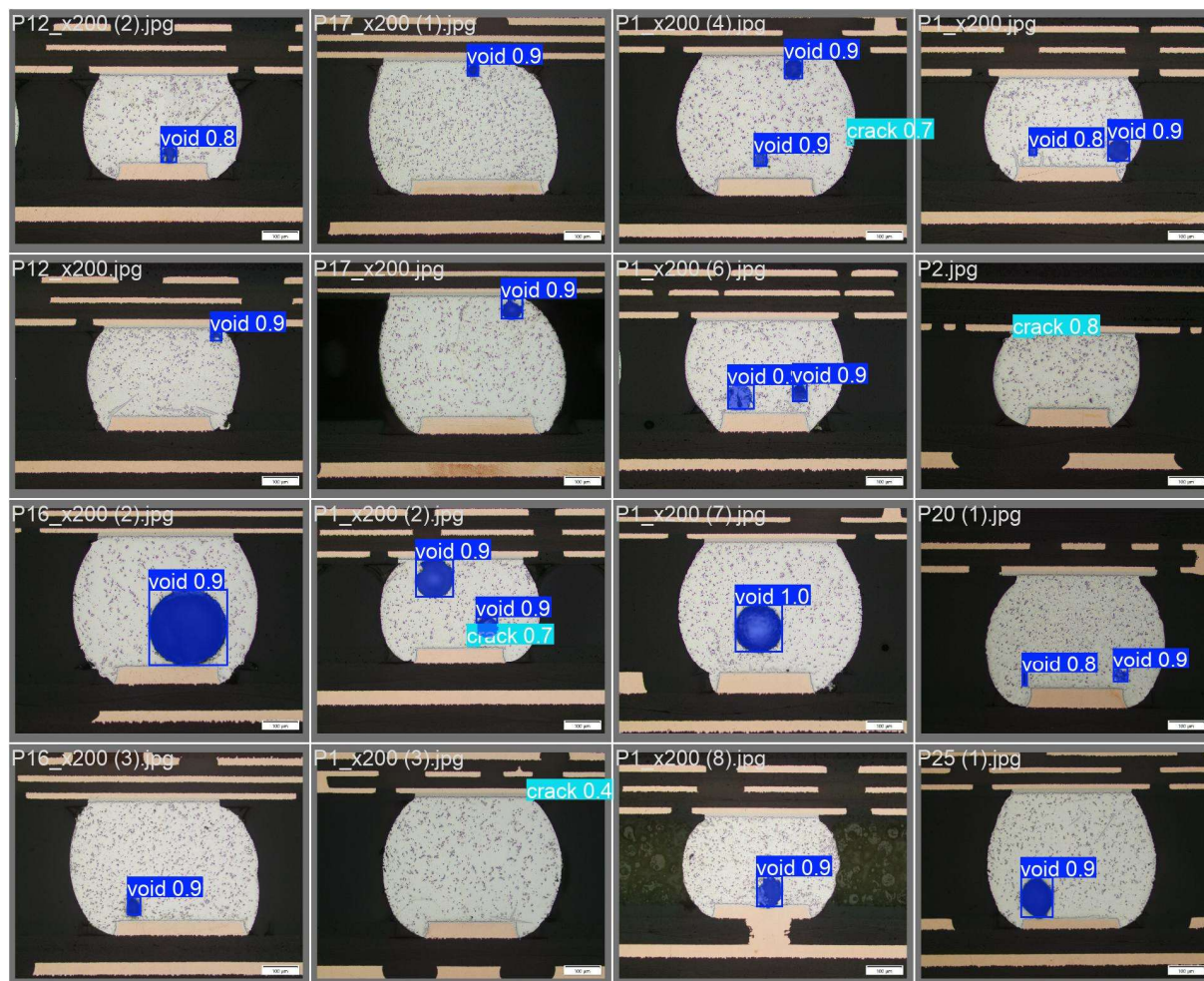
42. ábra Tréning köteg 2



43. ábra Tréning köteg 1

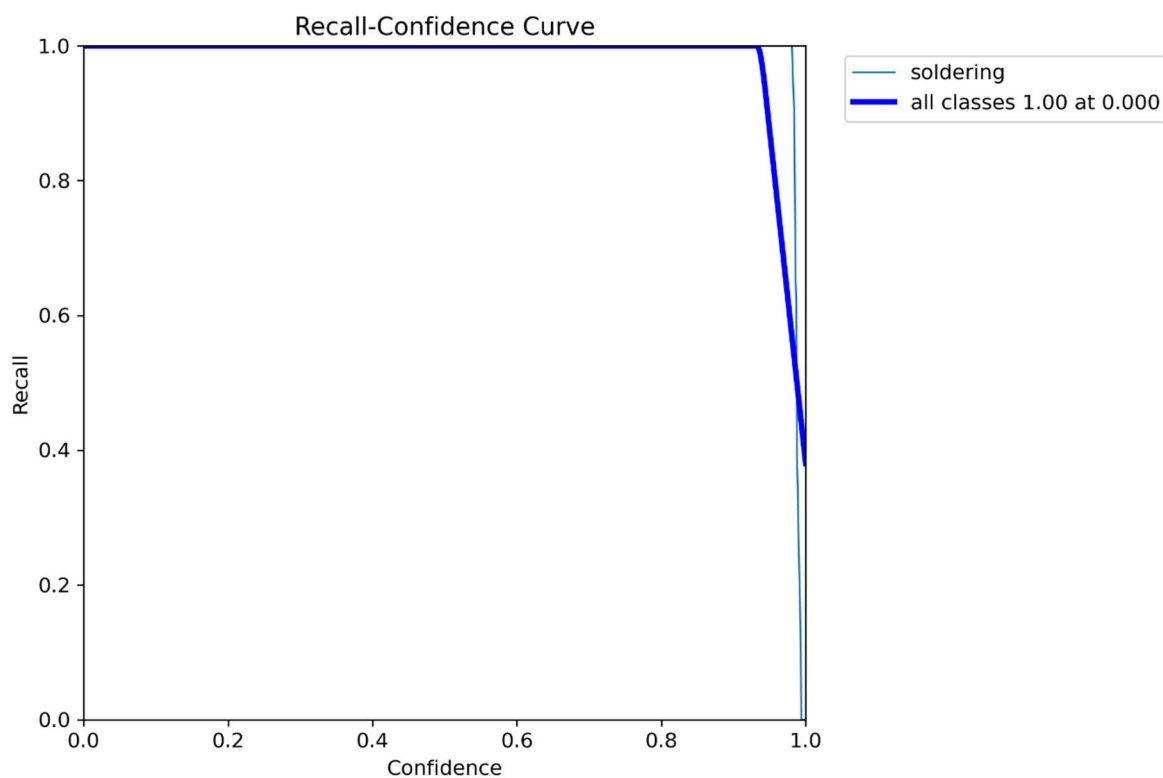


44. ábra Tréning köteg 0

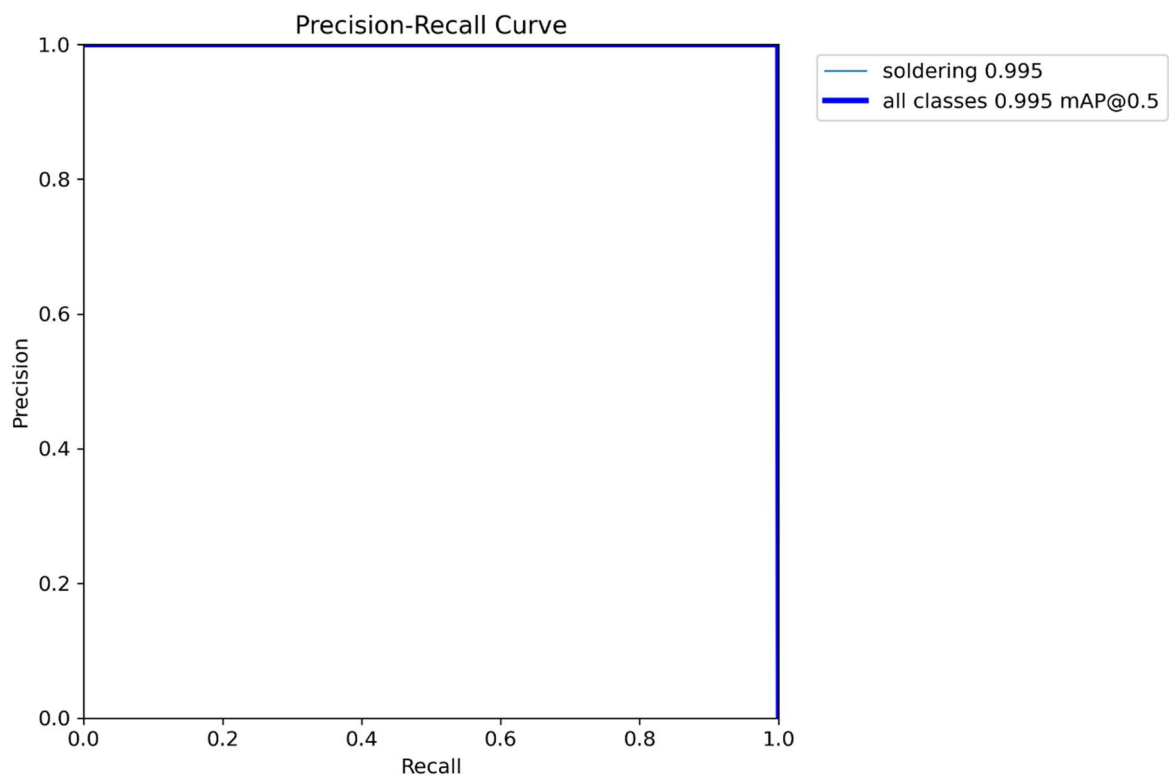


45. ábra Validációs köteg 2 - címkék

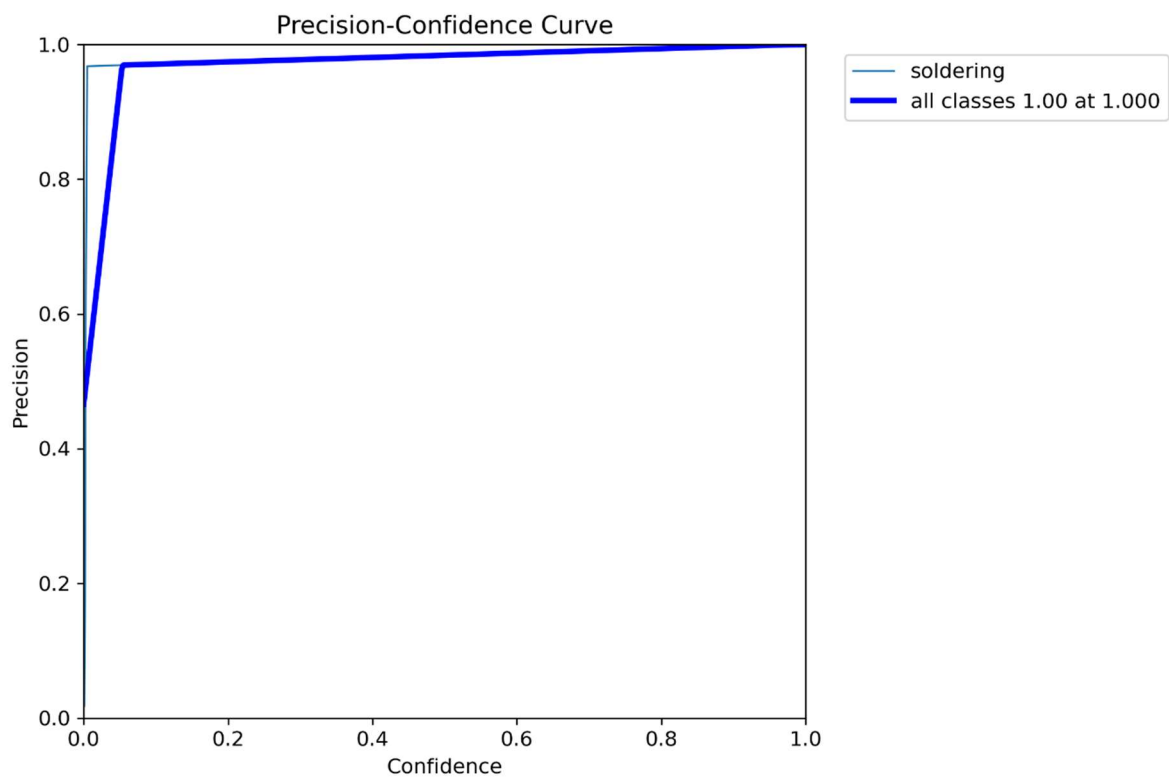
9.8 A második tanításhoz tartozó képek és diagrammok



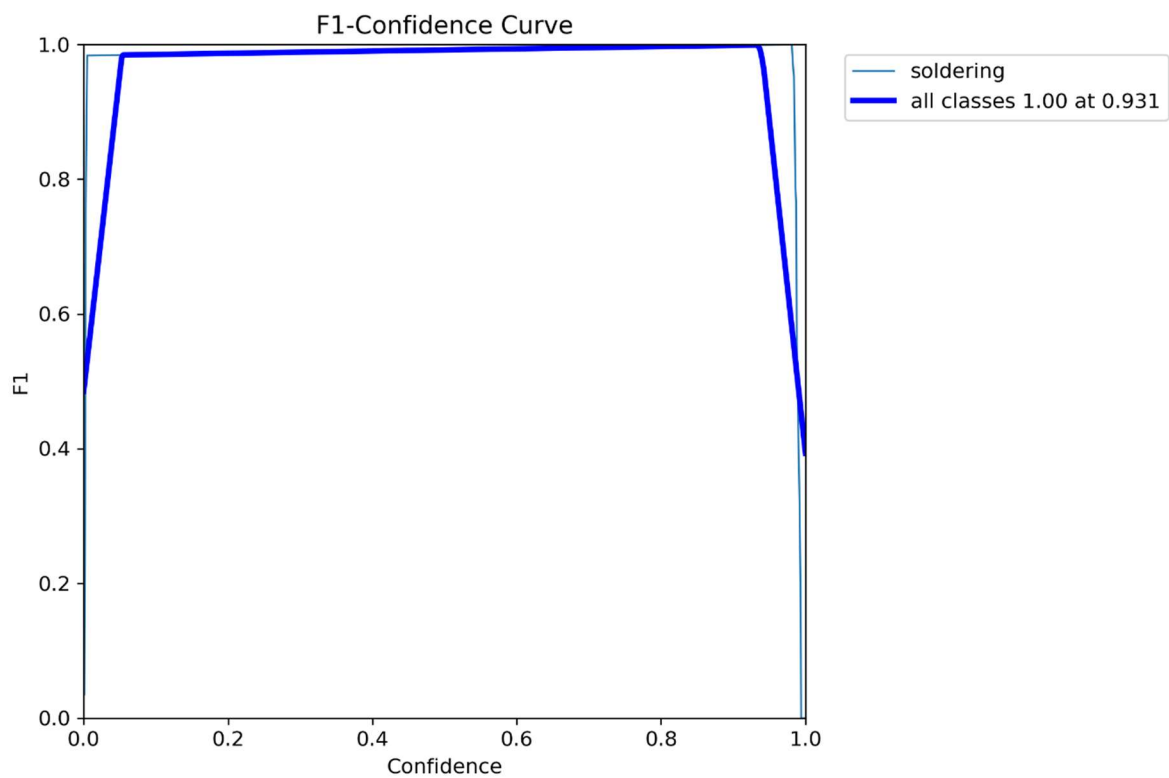
46. ábra Box Recall - Confidence diagram



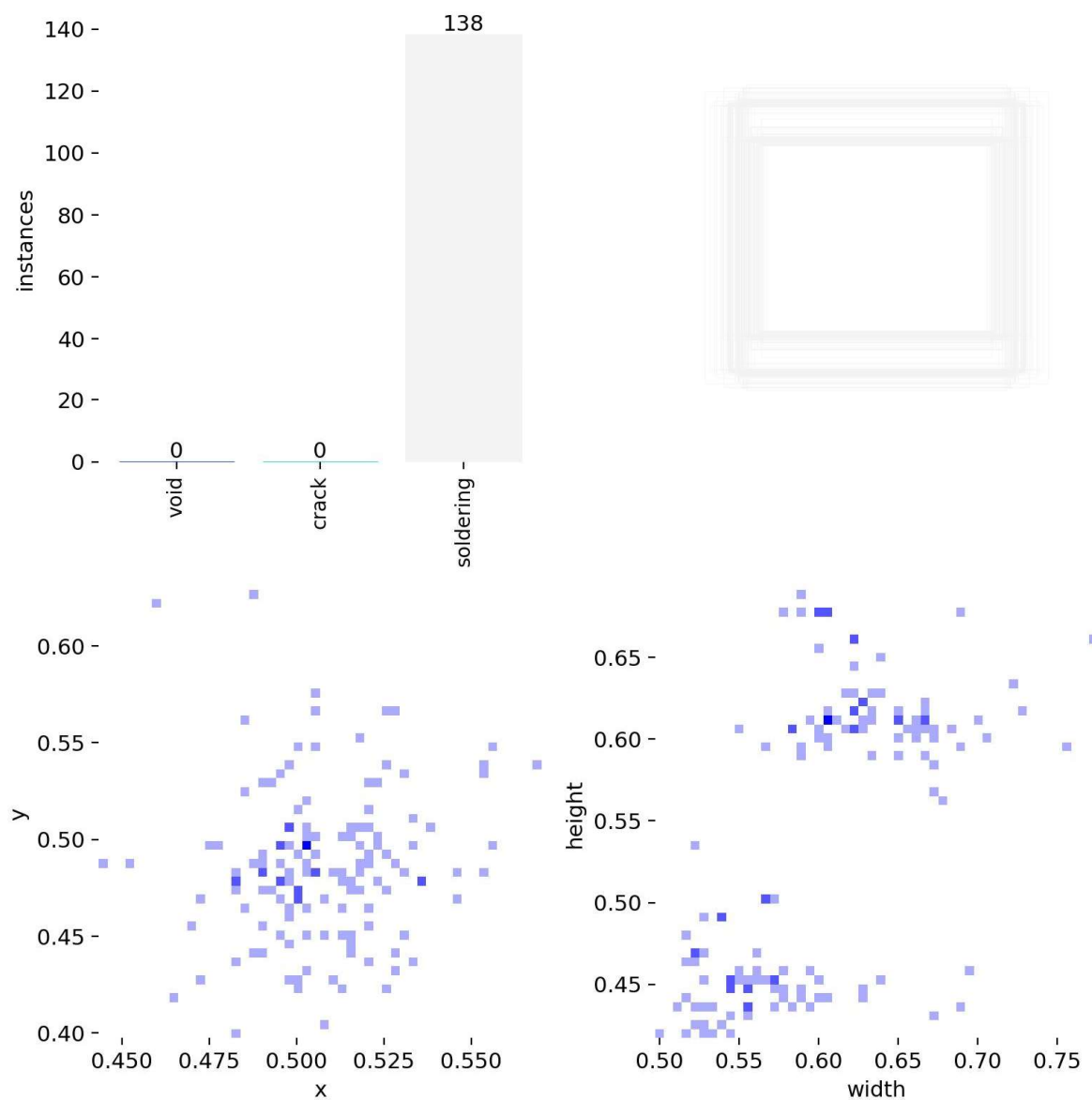
47. ábra Box Precision - Recall diagram



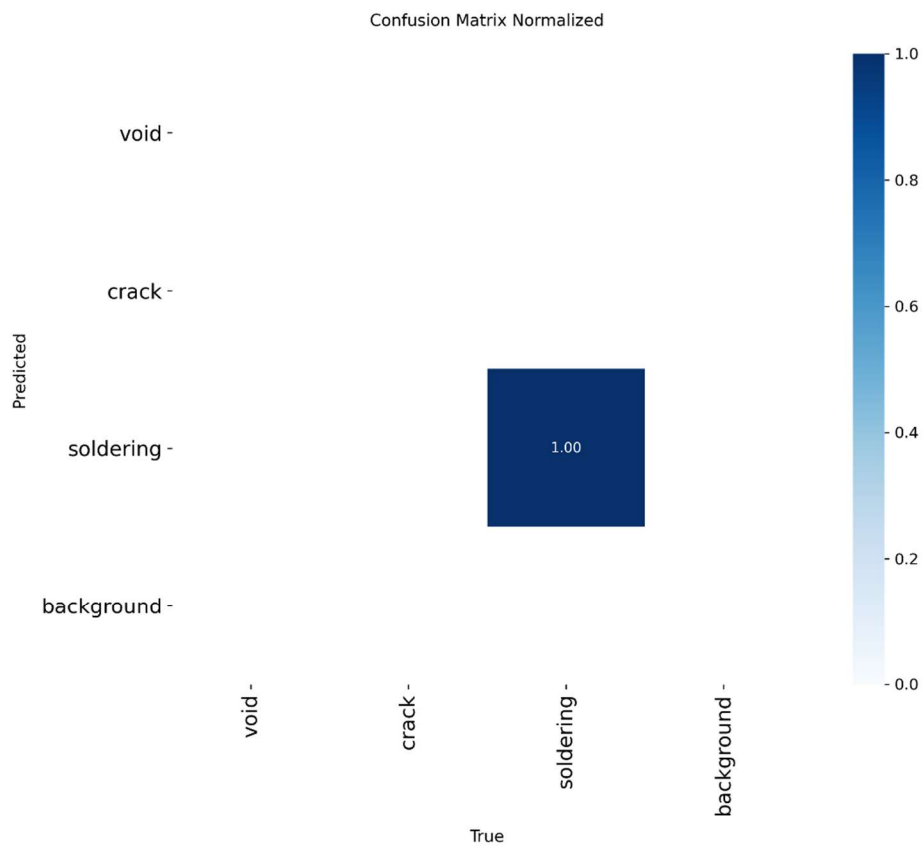
48. ábra Box Precision - Confidence diagram



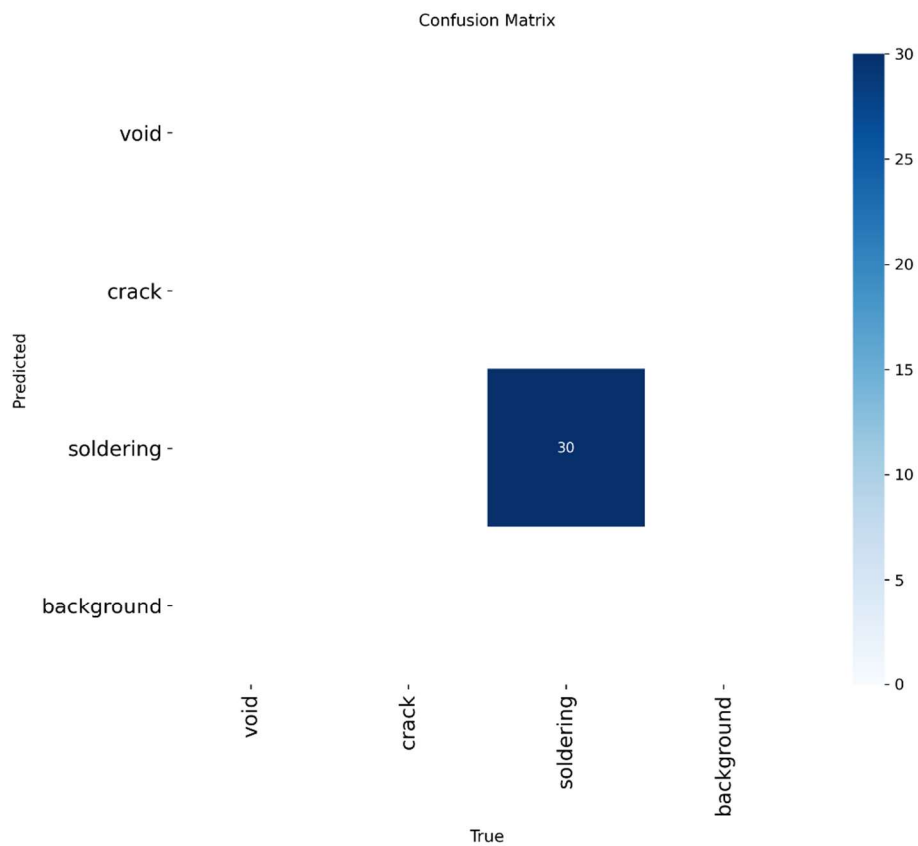
49. ábra Box F1 - Confidence diagram



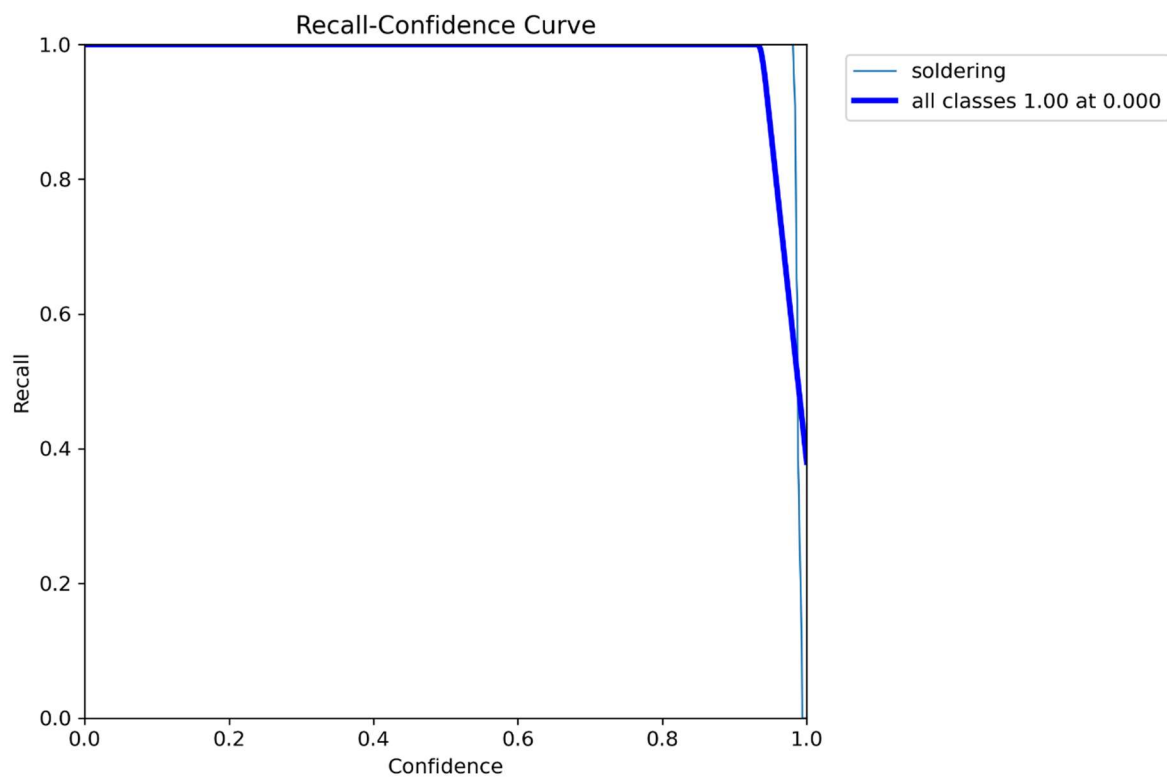
50. ábra Címkézési statisztika diagram



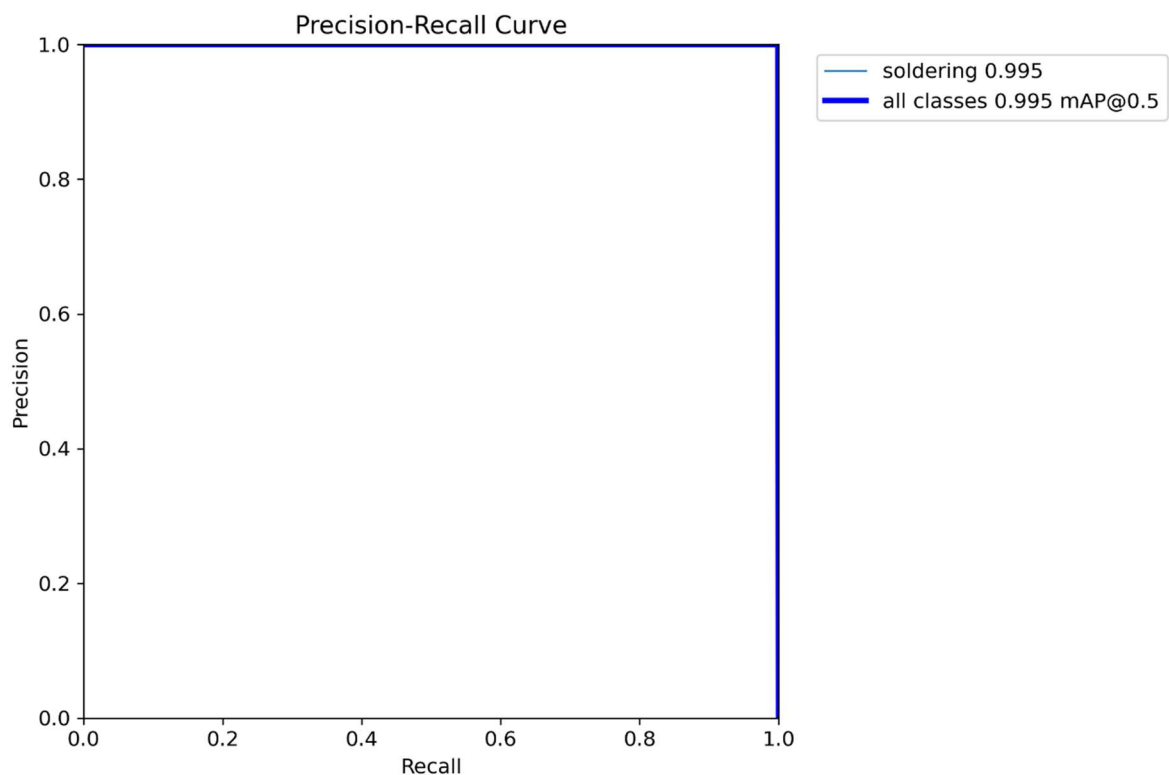
51. ábra Confusion Matrix Normalized



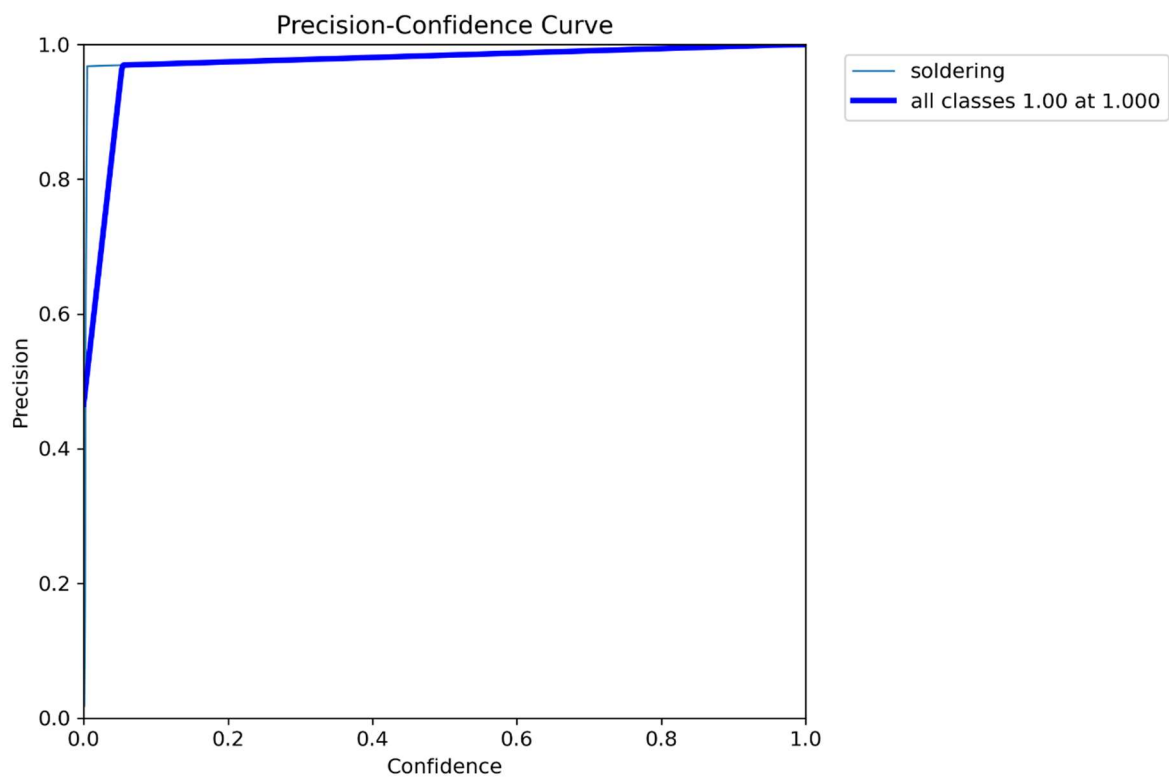
52. ábra Confusion Matrix



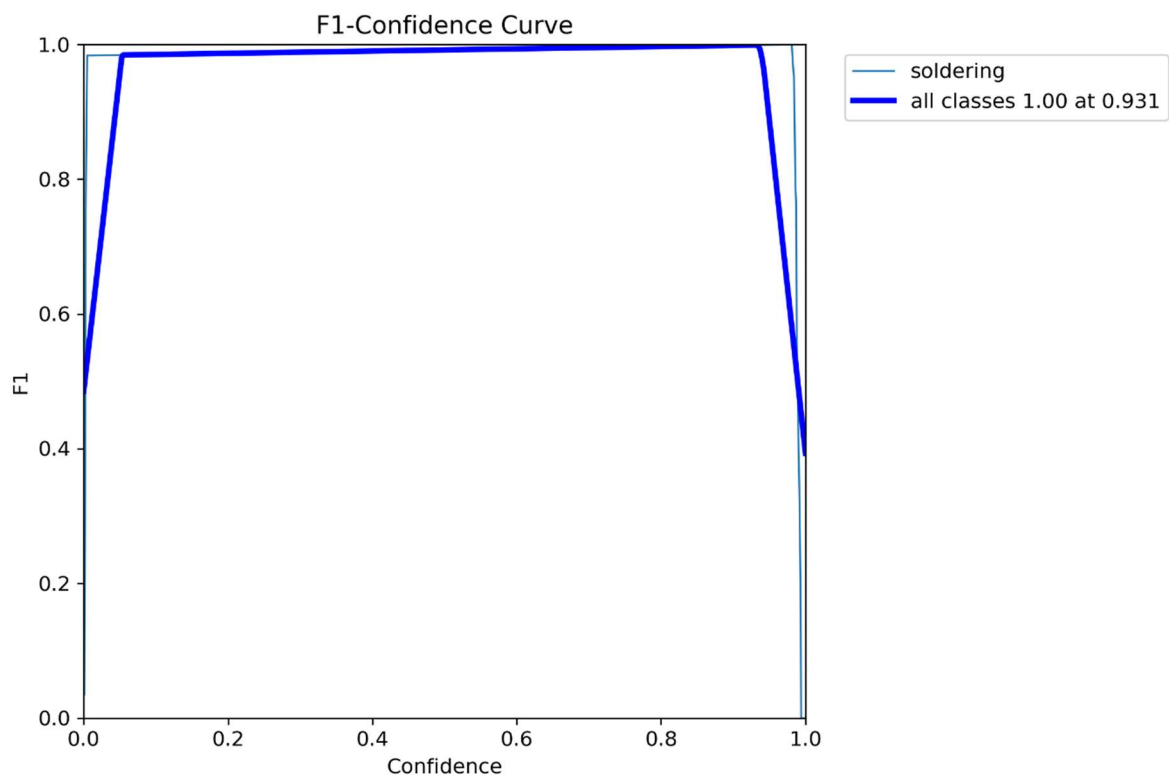
53. ábra Mask recall - Confidence diagram



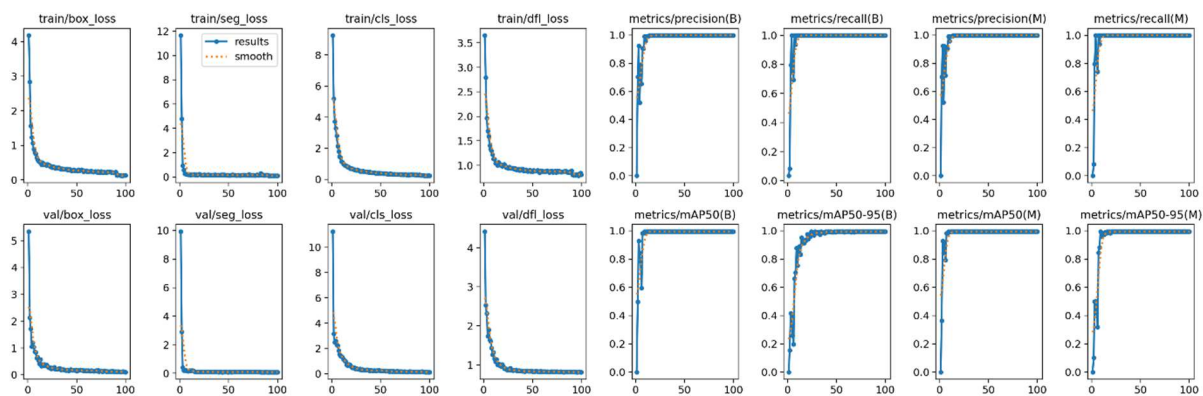
54. ábra Mask Precision- Recall diagram



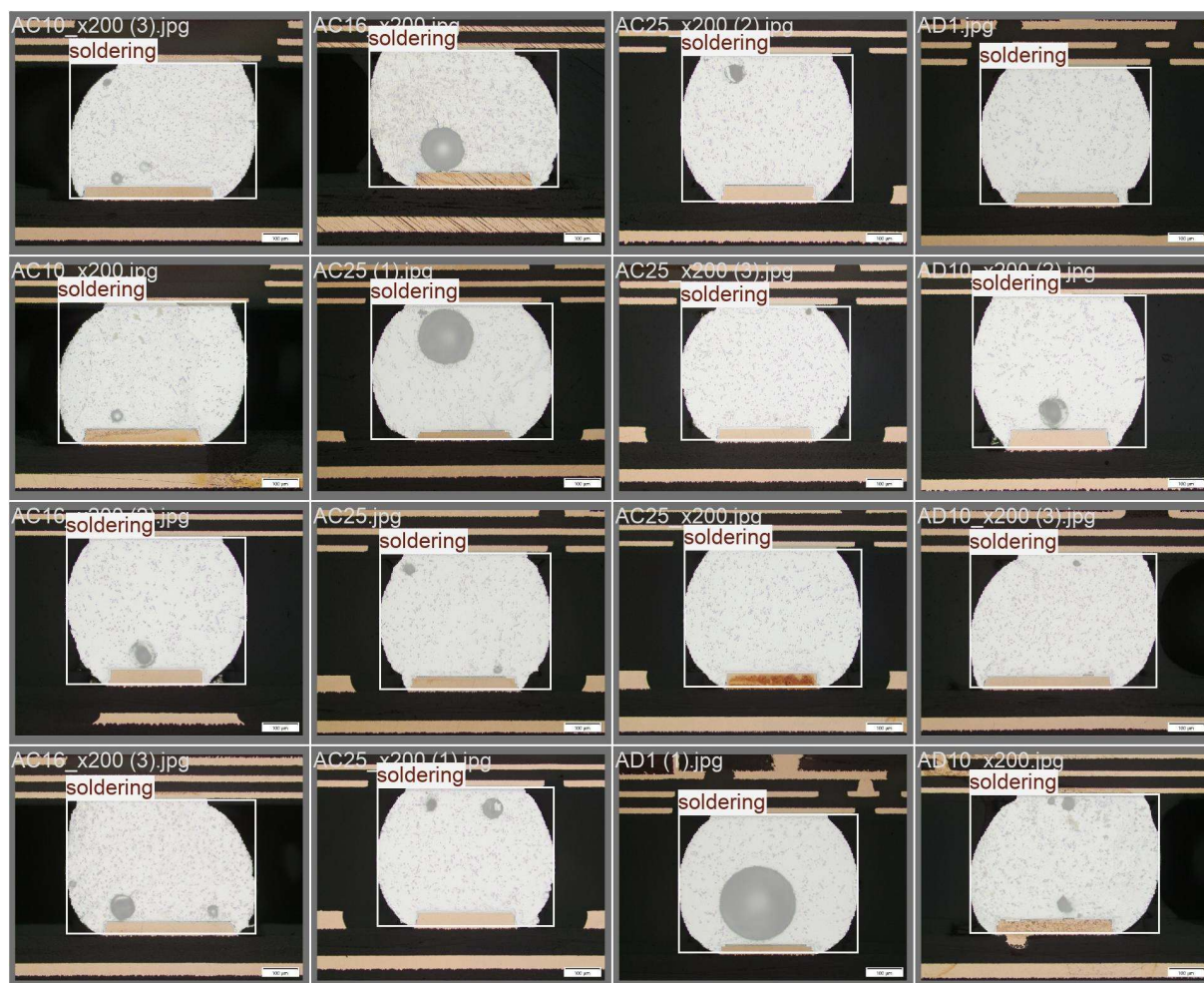
55. ábra Mask Precision - Confidence diagram



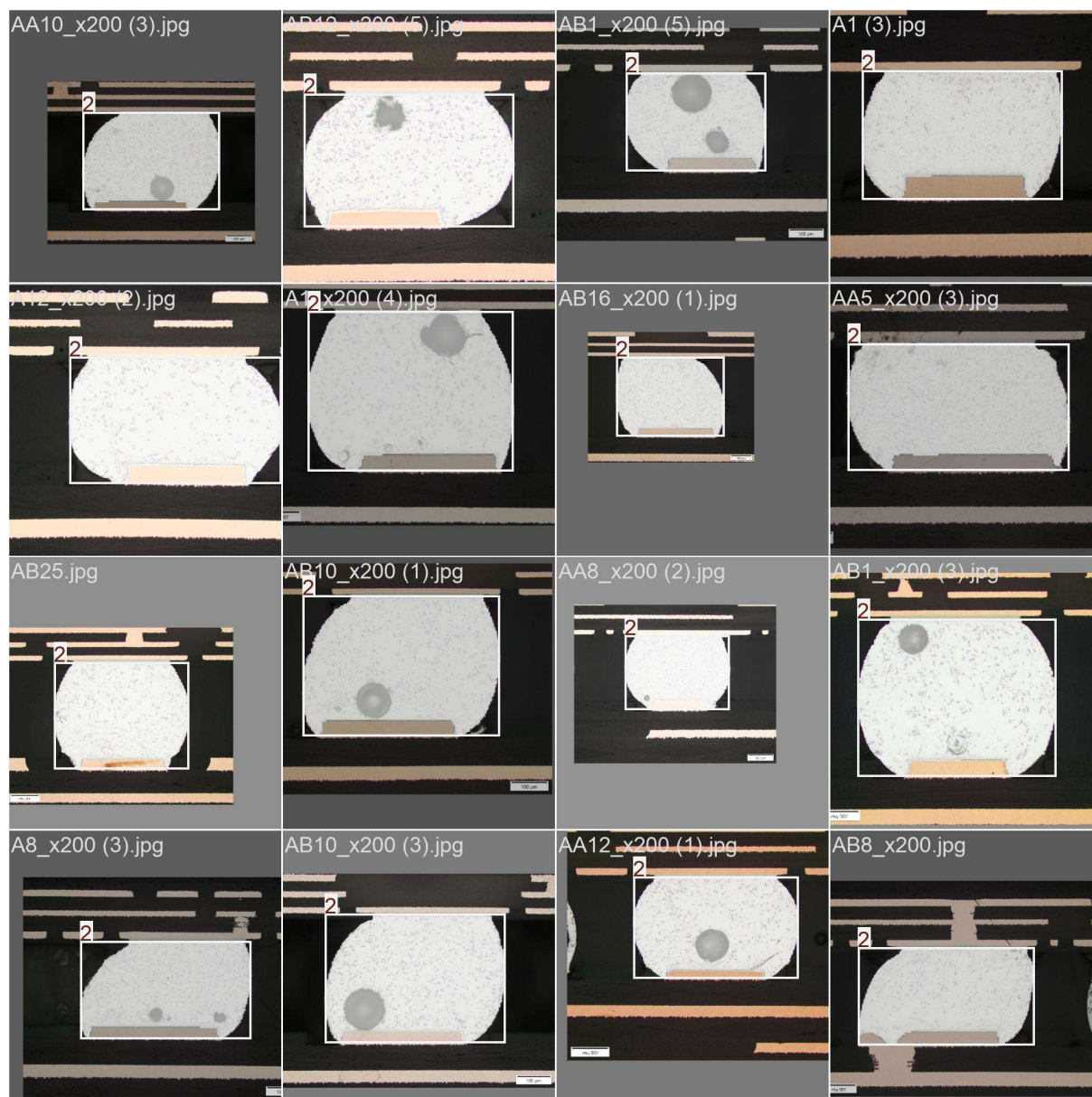
56. ábra Mask F1 - Confidence diagram



57. ábra Veszteség eredmények



58. ábra Validációs köteg 0 - Prediktált



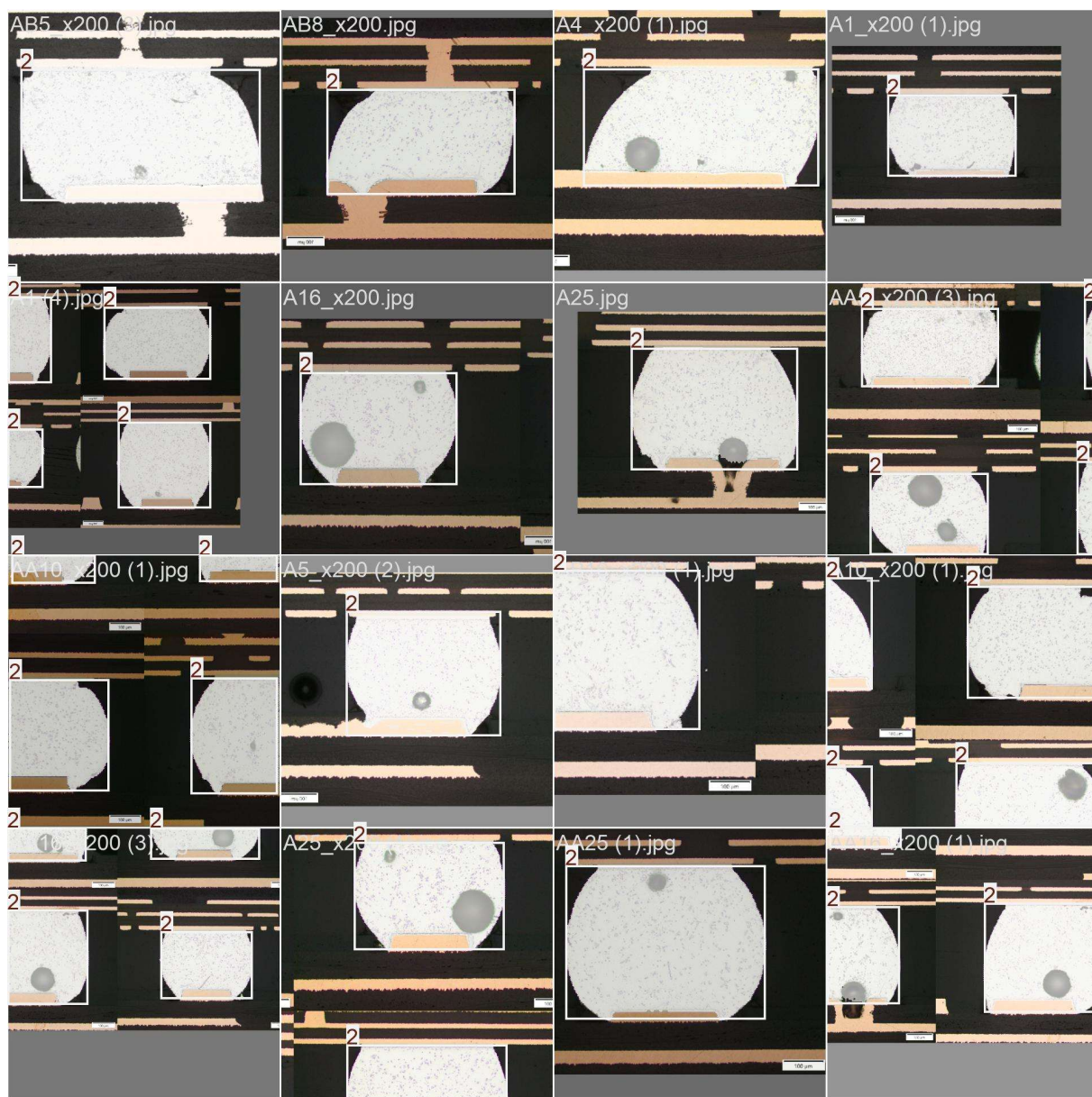
59. ábra Tréning köteg 2



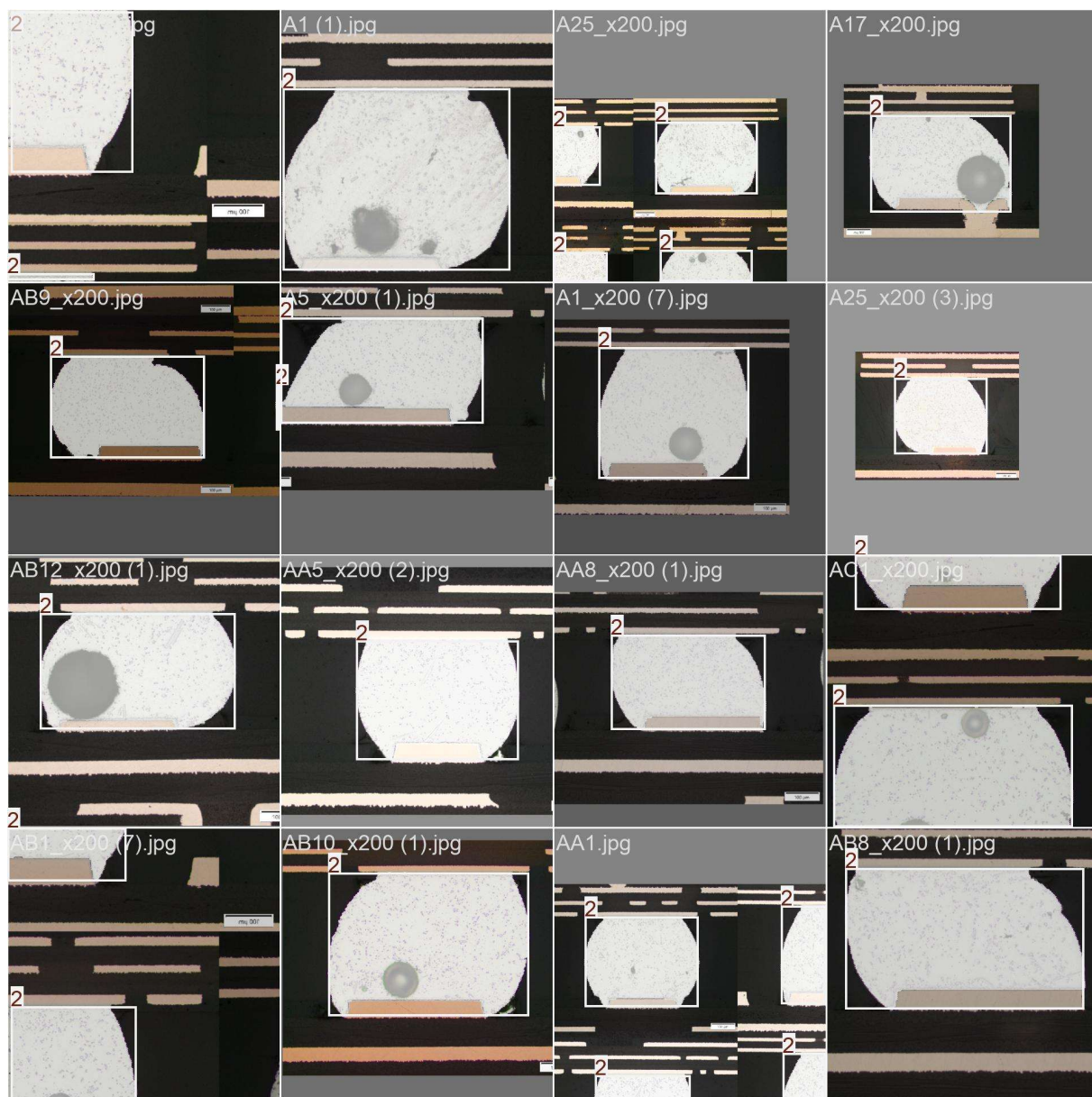
60. ábra Tréning köteg 1



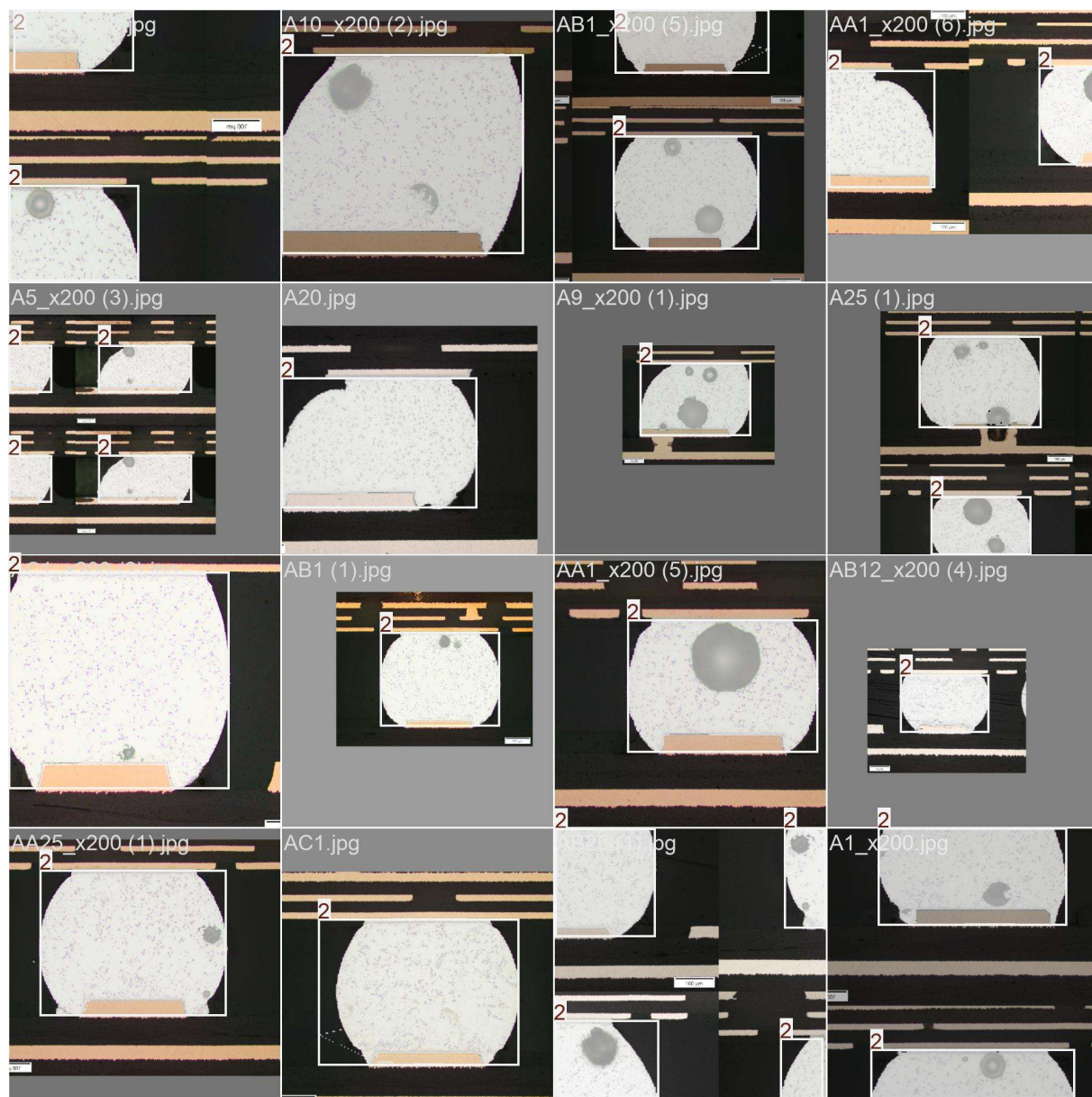
61. ábra Tréning köteg 0



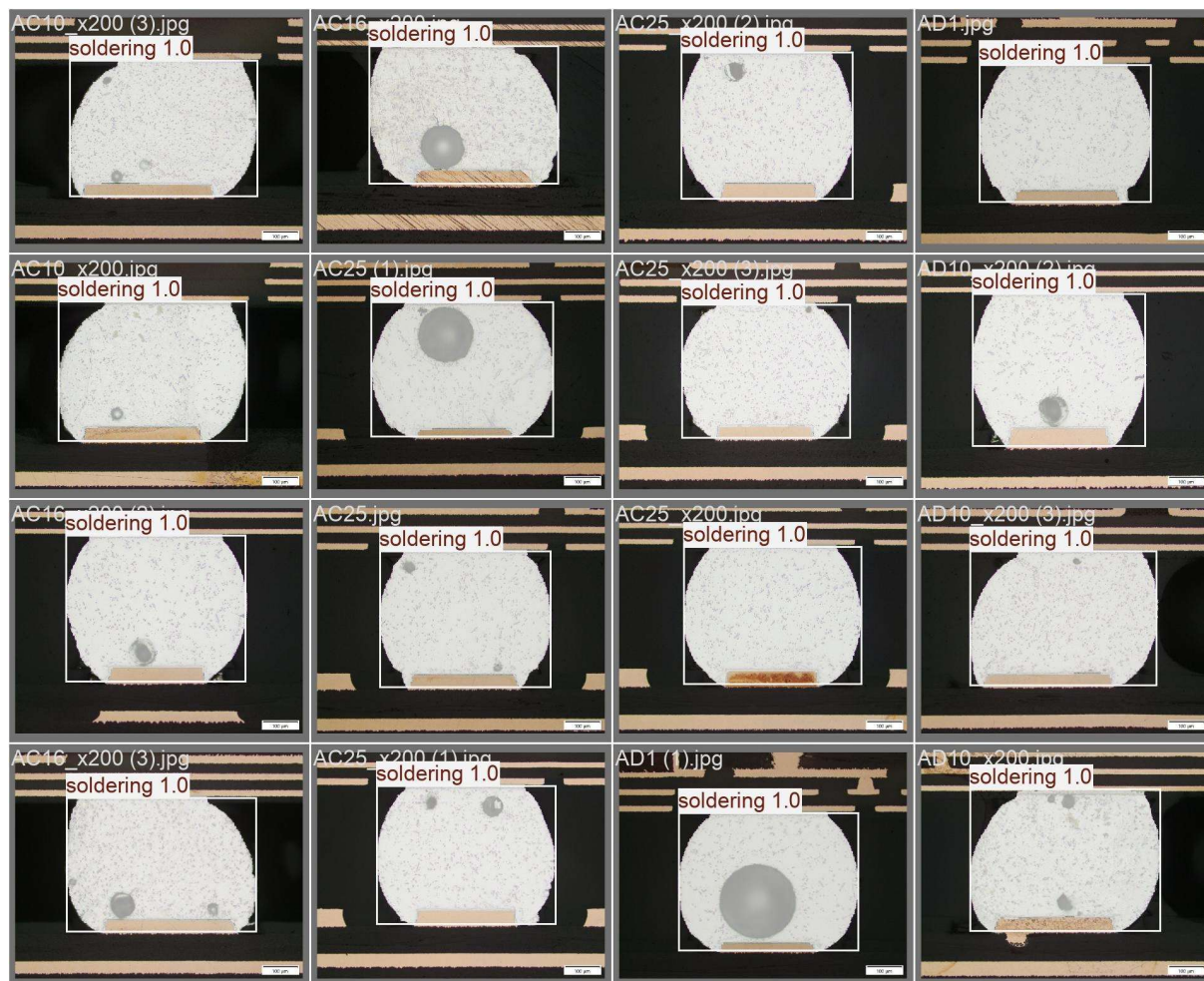
62. ábra Tréning köteg 2



63. ábra Tréning köteg 1



64. ábra Tréning köteg 0



65. ábra Validációs köteg 0 - Címkek