

TDK-DOLGOZAT

KÉSZÍTETTE: HOLICZA BARNABÁS

Holicza Barnabás

NEURÁLIS HÁLÓZATOK ALKALMAZÁSA A NAPELEM PANELEK MINŐSÉGELLENŐRZÉSÉRE

Témavezető: Dr.Vörösházi Zsolt, Konzulens: Amr Nagy Abdo

Műszaki Informatikai Kar, Villamosmérnöki és Információs Rendszerek Tanszék

Pannon Egyetem
Intézményi Tudományos Diákköri Konferencia
2022



Neurális hálózatok alkalmazása a napelem panelek minőségellenőrzésére

*Készítette: Holicza Barnabás
Programtervező informatikus BSc*

*Műszaki Informatikai Kar
Villamosmérnöki és Információs Rendszerek Tanszék*

Témavezető: Dr. Vörösházi Zsolt

Konzulens: Amr Nagy Abdo

Tudományos diákköri dolgozatomban az EL – Elektro-Lumineszcens technológiával a napelem panelekről speciális kamerával készült felvételek automatizált minőségellenőrzését tanulmányoztam és végeztem el.

A napelem panelek gyártása során különböző hibatípusokkal találkozhatunk, melyeknek azonosítása eddig emberi beavatkozással történt, mely szükségessé tette az gyors, és automatizált minőségellenőrzés folyamatának kidolgozását. Erre a problémára adok megoldást azáltal, hogy különböző mély-tanuláson alapuló neurális hálózatokat használok, mely modellek tanításával akár valós időben biztosítható a napelem panel hibáinak pontos detektálása és felismerése.

A neurális hálózatok tanításához egy EL napelem paneleket tartalmazó képi adatbázis állt rendelkezésemre, mely kézi annotálását elvégeztem és többször ellenőriztem. Az adatbázis strukturálása és későbbi bővítése során a DarkLabel/VIA annotációs programok segítségével kézi annotációk sorozatával kb. 1400 kép (~21000 hiba) annotációját sikerült meghatároznom, melyet a tanításokban sikeresen felhasználtam.

Az annotációk megjelenítéséhez, illetve a neurális hálózatok tanításához szükséges eltérő adatformátumok közötti konverziók egyszerűsítéséhez, valamint a detekciós eredmények megjelenítéséhez egy Python nyelven írt grafikus felhasználói felületet is implementáltam.

A neurális hálózatok (RetinaNet és YOLO) konfigurálását, tanítását és tesztelését különböző paraméterekkel is sikerült megvizsgálnom, melyek eredményeit különböző metrikák szerint a kiértékelés során ismertetem.

Kulcsszavak: neurális hálózatok, napelem panelek, Python, annotálás, hiba felismerés, EL vizsgálat

- [1] DEITSCH, Segmentation of photovoltaic module cells in uncalibrated electroluminescence images. *Machine Vision and Applications*, Springer, **2021**, 32.4: 1-23.
- [2] MENG, Defect object detection algorithm for electroluminescence image defects of photovoltaic modules based on deep learning. *Energy Science & Engineering*, Wiley, **2022**.

Köszönetnyilvánítás

Ezúton szeretnénk köszönetet mondani a 2019-1.1.1- PIACI-KFI-2019- 00323 *"Különböző építőanyagokba integrálható, innovatív kialakítású napelem panelek gyártásfejlesztése, magas fokon automatizált, adaptív gyártási technológia kialakításával"* c. projekt keretében az NKFI Alap anyagi támogatását, a Magyar Kutatási Alap OTKA K-135729 számú pályázatának anyagi támogatását, valamint az NVIDIA GPU Grant Program (2018) hardver eszköz támogatását.

Szeretnék köszönetet mondani témavezetőmnek, Dr.Vörösházi Zsoltnak, hogy figyelemmel kísérte a tudományos diákköri dolgozatom készülését, és hasznos tanácsaival, motivációjával, és segítőkészségével nagyban támogatta a dolgozatom elkészülését.

Továbbá szeretnék köszönetet mondani konzulensemnek, Amr Nagy Abdonak, hogy a neurális hálózatokkal kapcsolatos tapasztalataival megkönnyítette a munkámat, és elősegítette tudásával a gépi tanulási ismereteim bővítését.

Tartalomjegyzék

1. Bevezetés	7
2. Napelem panelek tulajdonságai	11
2.1. Napelem panel működése	11
2.2. Hibatípusok	13
3. Adatbázis	17
3.1. ELPV	17
3.2. TecnaliaPR	18
3.3. ELDDS - saját adatbázis	19
4. Képek előfeldolgozása	22
4.1. Hisztogram transzformációk	22
4.2. Wallis-szűrő	23
4.3. CLAHE	24
5. Annotálás és detektálás	27
5.1. Mérési környezet	27
5.2. Annotáló eszközök és a formátum átalakító alkalmazás	30
5.2.1. DarkLabel	30
5.2.2. VIA annotációs program	32
5.2.3. Solaris formátum átalakító program	33
6. Neurális hálózatok	36
6.1. Neurális hálózatok általánosan	36
6.1.1. Egy-, és kétlépcsős detektor hálózatok	37
6.2. Alkalmazott neurális hálózatok	39
6.2.1. RetinaNet hálózatok	39

6.2.2. YOLOv3 és YOLOv5	40
7. Eredmények kiértékelése	42
7.1. Hibastatisztika	42
7.2. Fogalmak, metrikák	43
7.3. Eredmények	46
7.3.1. Első annotálás (~ 1300 db kép) utáni eredmények	46
7.3.2. ELDDS-I: részleges annotálás (~ 620 db kép) utáni ered- mények	47
7.3.3. YOLOv5 modellek vizsgálata	48
7.3.4. Modell paramétereinek vizsgálata és beállítása	49
7.3.5. Veszteségfüggvény módosítása	50
7.3.6. ELDDSI+II: teljes annotálás (~ 1300 db kép) utáni ered- mények	53
8. Összefoglalás	55
Irodalomjegyzék	57

1. fejezet

Bevezetés

Magyarországon az utóbbi években a napelem rendszerek használata egyre népszerűbb lett, mint a legkorszerűbb megújuló energiaforrás az ipari, illetve családi háztartások területein. Ez a terjedés azt eredményezte, hogy az ország teljes áramtermelésének mintegy 10.6%-át már a napenergia biztosítja (2021-es adatok alapján) [1]. Ez a térhódítás a napelemek tömeges gyártását eredményezte. Ez természetesen minőségellenőrzéssel párosul, azonban a megnövekedett igények és a folyamatosan növekvő napelemszám miatt az emberi-vizuális vizsgálat már szinte lehetetlen. Annak érdekében, hogy a gyártó cégek a követelményeknek mindenben megfeleljenek, a gyors és automatizált termelés és minőségbiztosítás egy fontos szempontjává vált a minőségi panelek előállításának. A cellákon található, általában gyártásból és külső tényezőkből eredő hibák gyakran nagyon nehezen láthatóak, azonosításuk időigényes, mely a profit szempontjából a cég számára kiesést okozhat, mivel a napelem minősítése határozza meg a panelek kategorizálását. Ezért a cégek igényt tartanak a napelemek gyors, pontos és automatizált bevizsgálására. Jelen dolgozatban ennek a minőségellenőrzési folyamatnak a támogatására egy EL-tesztelő berendezés hiba-detekciós-, és felismerő részét kívánom megvalósítani, melyhez a neurális hálózatok információfeldolgozó képességét használom.

Honnan származik a neurális hálózatok ötlete? A mesterséges neurális hálózatok (ANN - Artificial Neural Networks) ötlete az 1980-as évekből származik, melyet a különböző időigényes folyamatok automatizálására kifejlesztett technológiák optimalizálására kívántak használni. Nem meglepő módon az agyban található neuronok (idegsejtek) működésének modelljét használták fel: „konnekcionizmusnak” nevezve az áramkörök olyan módú összekapcsolását értették, amely valamilyen intelligens viselkedést eredményezett, vagy szimu-

lált. Ezen hálózatok az emberi agy neurális kapcsolatrendszerének feldolgozási képességét és emlékezőképességét kívánják reprodukálni különböző algoritmi-
kus technikákkal. Az elmúlt 10 év sikerei is a különböző neurális hálózatok hatékonyságát tanúsítják, melyet az ipar és az egészségügy különböző területei jól demonstrálnak: mintafelismerés, robotika, automatizálás; biológiai-, gazdasági-, gyógyszerészeti kutatások támogatása.

1943-ban Warren McCulloch amerikai pszichológus és Walter Pitts matematikus létrehozta egy olyan algoritmust, amely az általuk kitalált egyszerű "M-P modellt" használta, amely funkcionális logikai eszközökként kezelte a neuronokat. A modell működését egy egyszerű elektromos áramkörrel próbálták demonstrálni. Ez volt a neurális hálózatok kutatásának első nagy lépése. 1949-ben Donald O. Hebb pszichológus a „The Organization of Behavior” című könyvében hipotézist állított fel arra vonatkozólag, hogy az idegsejtek szinapszisainak (kapcsolatainak) intenzitásai változó mértékben vannak jelen. Javaslatát az volt, hogy az idegi pályák minden egymást követő használat során erősödjenek, különösen az olyan neuronok között, amelyek hajlamosak egyszerre „tüzelni”, aktiválódni [2]. A tanítási fázis a neuronok között húzódó szinapszisok felületén történik. Ezt a folyamatot vizsgálva Hebb rájött arra, hogy a idegsejtek összeköttetései közötti intenzitás változik, köszönhetően a szinapszis előtti és utáni neuronoknak. Ez alakította ki a híres "Hebb- szabályt", a neurális hálózatok tanításának és az emlékezőképességének alapját, amely azt mondja ki, hogy két neuron közötti súlytényező erősségét milyen mértékben kell növelni, vagy éppen csökkenteni. Két fő koncepció alakult ki az 1940-es években, amelyeket a neurális hálózat előljáróinak tekinthetünk: „Threshold Logic” – a folyamatos bemenet átalakítása diszkrét kimenetűvé, „Hebbian Learning” – az idegi formálhatóságon alapuló tanulási modell, melyben azok a sejtek, amelyek egyszerre „tüzelnek”, azok össze is kapcsolódnak. Az 1950-es években a kutatók ezeket a módszereket próbálták számítógépes környezetbe átültetni, és 1954-ben megjelent az első Hebbian-féle hálózat az (MIT). 1957-ben megjelent Frank Rosenblatt által publikált Perceptron modell, amely az M-P modell és egy rovar (légy) szemében rejlő „döntési” rendszerek alapján készült. Ez a modell a neurális hálózatok alappilléreinek tekinthető, hiszen szerkezete kapcsolatban áll a neurofiziológiával. Ennek a modellnek a fő előnye a folytonosan változtatható súly. Ezek tudatában Rosenblatt a kutatásaival bebizonyította

a kétszintű (bemenet-kimenet) szenzorok osztályozási lehetőségeit, és egyben korlátaikat, valamint célkitűzést tett egy háromrétegű architektúra megvalósítására (rejtett rétegek segítségével). Warren S. McCulloch idegtudós és Walter Pitts javaslatára az agyi döntési folyamatok szimulálására egy lineáris rendszert alakítottak ki, amelyben a súlyozott összegek vagy '0'-át, vagy '1'-et adnak eredményül. 1959-ben B. Widrow és M. Hoff mérnökök a legnagyobb áttörést jelentő adaptív lineáris tényező (ADALINE) alkalmazásával álltak elő. Létrehozták a Windrow-Hoff tanítási szabályt, amely a legkisebb négyzetes eltérés módszerén alapszik. Ez a módszer annyira hasznosnak bizonyult, hogy a mind mai napig sikerrel alkalmazzák.

1972-ben Kohonen T. finn professzor kidolgozta a "Self-Organizing Feature Map" módszerét. Ezen technológia segítségével a mintafelismerés, a beszéd felismerés és az osztályozási problémákra fókuszáltak.

1982-ben John Hopfield amerikai fizikus létrehozott egy diszkrét neurális hálózatot (Hopfield néven), melyben az energia függvény alkalmazásának első gondolata vetődött fel, amely segített meghatározni az adott neurális hálózat stabilitását. Ezt a függvényt Lyapunov függvénynek nevezte el. A kutatások itt nem értek véget, tovább folytatta a neuronok szerkezetének vizsgálatát, a rejtett rétegek különböző alkalmazhatóságának teszteléseit is. 1985-ben az ún. (TSP - Traveling Salesman Problem) "utazó ügynök problémára" kiélezve sikerrel alkalmazták a neurális hálózatot. Hopfield hálózatának hatékonyságát a nem-lineáris differenciálegyenletek használata jelentette, amely segített az információ tárolásában és a visszacsatoló függvény eredményes alkalmazhatóságában. Ezen eredményeknek köszönhetően Hopfield megosztotta neurális hálózati modelljét, főleg olyan célból, hogy akár egyetemi kutatások által gyorsabb fejlesztést tudjon elérni, újabb ötletekkel tudja módosítani a hálózatát. 1983-ban Kirkpatrick az ún. SA - simulated annealing (szimulált lágyítás) globális optimalizálási algoritmusát felhasználva NP-teljes problémák megoldását segítették. Ahogy fejlődtek a neurális hálózatok be kellett látni, hogy minél komplexebb a modell architektúra, annál nagyobb erőforrás igényel is rendelkezik. Ebből kifolyólag 1984-ben Hilton statisztikai-fizikai tulajdonságokat alapul véve előállt a neurális hálózatok párhuzamosításának ötletével, amelyet később Boltzmann rendszernek neveztek el. Ebben a rendszerben a neurális hálózat tanítási folyamatában a párhuzamos tanítás feltételeként a

több-rétegű tanítást helyezte előtérbe. 1988-ban (Chua-Yang-Roska) a Celluláris Neurális/Nem-lineáris Hálózatok (CNN) megjelenésével a nagy számítás-igényű tanítási folyamatok támogatására adtak lehetőséget. 1991-ben Haken a kognitív folyamatot támogató, véletlenszerűséget kihasználó együttműködést fedezett fel a neurális hálózatokban. A neurális hálózatok fejlődésének alapját a matematika fejlődése is segítette, hisz ezen folyamatok komplex egyenletek sorozatával írhatóak le. Ezekután mindinkább feladat-specifikus hálózatok tervezése és megvalósítása következett, mindet más-más felhasználásra tervezve. Különbségek csak a felhasznált tervezési-fejlesztési technológiákban voltak. [3]

Napjainkban nagyon sok neurális hálózatot publikálnak, melyek a különböző régi és korszerű algoritmusokat (például a K-mean) felhasználva több terület időigényes folyamatait támogatják. A virágzást az újabb mesterséges intelligenciához köthető technológiák jelentették, mint például a 2006-ban megjelent mély-tanuláson alapuló DNN - (Deep Neural Network) hálózatok, amelyek rejtett rétegek sokaságát tartalmazzák (akár több száz réteg), mellyel nagy mennyiségű információt lehet kinyerni komplex adathalmazokból.

A feladatom során a napelem paneleken található hibák detektálását és felismerését vizsgálom modern képfeldolgozási módszerekkel, különböző neurális hálózatok segítségével, melyek tesztelése és kiértékelése megfelelő háttérrel biztosít a gyors és automatizált vizuális minőségellenőrzéshez.

2. fejezet

Napelem panelek tulajdonságai

2.1. Napelem panel működése

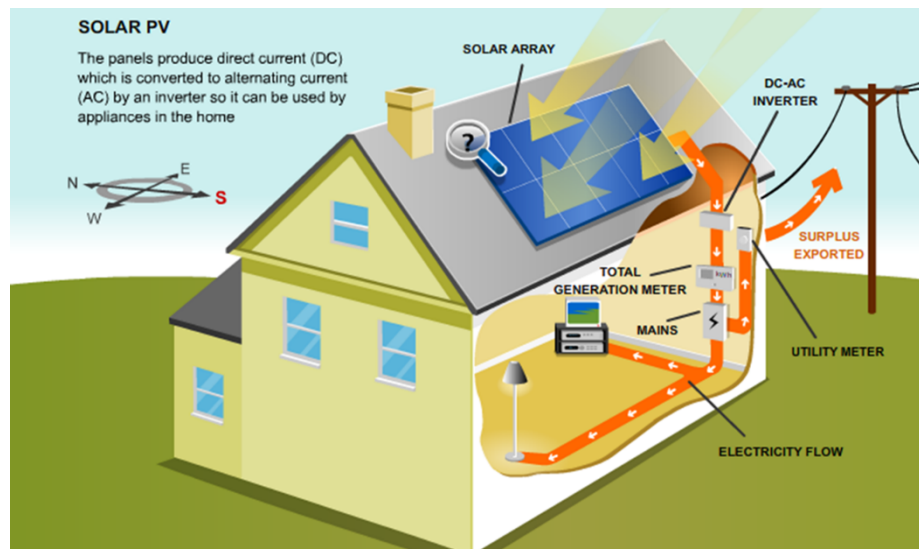
A napelemek, vagy más néven (photo-voltaic) PV-panelek, a napfényt elektromos árammá átalakító eszközök, melyek megújuló energiaforrásként használhatóak fel az élet számos területén, pl.: a háztartásokban, intézményekben, kültéri világításban, stb. De hogyan is működnek ezek a napelemek, hogyan alakítják át a napfényt elektromos energiává?



2.1. ábra. Tetőcserépre integrált Domino napelem panel - CorecommSI Kft.

Egy napelem panel sok kis foto-voltaikus cellából áll, melyek az anyagi tulajdonságában rejlik a napfény villamos energiává átalakítása. Ezek a cellák szilárdtest félvezető anyagból készülnek, leggyakrabban szilíciumból, amely képes vezetni az elektromosságot, miközben az elektromos mező létrehozásához

szükséges elektromos egyensúlyhiányt fenntartja. Amikor a napfény eléri a napelem félvezetőjét, a fényből származó energia fotonok formájában elnyelődik, ezen kívül számos elektront átenged, amik aztán szabadon sodródhatnak a cellában. A napelemet pozitív-, és negatív töltésű félvezetőkkel tervezik, amelyek elektromos mezőt hoznak létre. Ez az elektromos tér arra kényszeríti az elektronokat, hogy bizonyos irányban áramoljanak – a cellát szegélyező vezetőképes fémlemezek felé. Ezt az áramlást energiaáramnak nevezzük, és az áram erőssége determinálja, hogy az egyes cellák mennyi villamos energiát tudnak termelni. Ahogyan a szabad elektronok eléri a fémlemezeket áram indukálódik a vezetékben. Amikor a napelemek villamos energiát termelnek, az energia egy sor vezetéken keresztül egy áram átalakítóhoz, az ún. inverterhez áramlik. Bár a napelemek egyenáramot (DC) termelnek, a legtöbb villamosenergia-fogyasztó váltóáramot (AC) igényel, pl. az épületek ellátásához. Az inverter feladata, hogy az áramot DC-ről AC-ra alakítsa a napi használat során. A tápfeszültséggé átalakított (DC→AC) elektromos áramot az inverter a megszakító dobozhoz továbbítja [4].



2.2. ábra. Napelem panelek működése a háztartásokban

Elektromos áram előállításának fontosabb lépései:

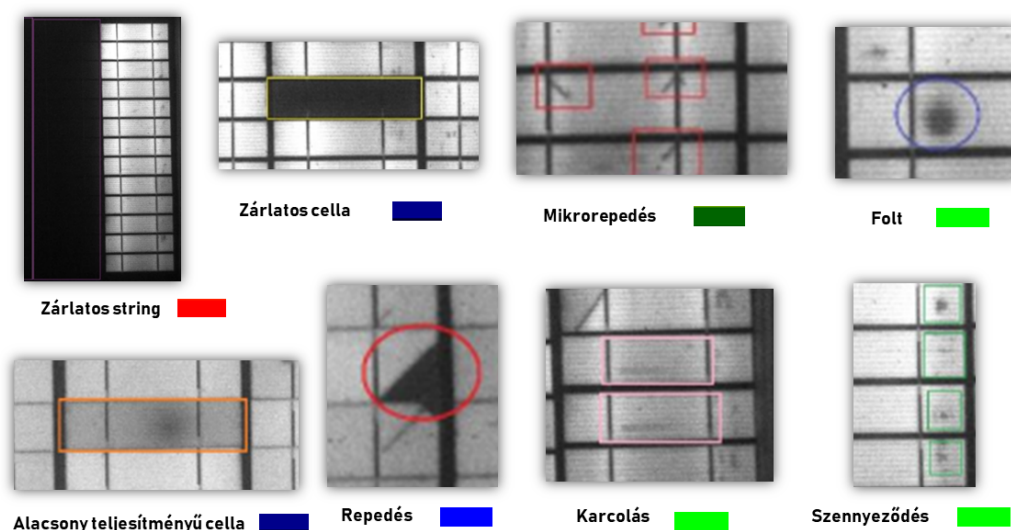
1. Amikor a napfény (elektromágneses sugárzás) eléri a napelem cellákat, és elektromos mezőt hoz létre.

2. A generált áram a panel széléhez jut a cellákat összekötő vezetőképes fémzárlakon ("busbar") keresztül.
3. A vezeték táplálja az invertert, amely az egyenáramot váltóárammá (DC-AC) alakítja az elektromos hálózatba kapcsolható.
4. Egy másik vezeték váltóáramú tápellátást biztosít az invertertől az ingatlan kapcsolótáblájáig (megszakítódoboznak is nevezik), amely szükség szerint elosztja az áramot az egész épületben.
5. Amennyiben a megtermelt villamos energiát nem fogyasztjuk el, úgy egy mérőórán keresztül a nyilvános hálózatba áramlik (termel). Amikor a megtermelt áram átfolyik a mérőn, a mérő fordítva működik, és jóváírja a többlettermelést. [5]

2.2. Hibatípusok

A napelem panelek gyártása során különböző hibák fordulhatnak elő. A feladat kiírásakor a detektálást és minősítést igénylő napelem gyártó (CorecommSI Kft.) elsőként 10 különböző hibaosztályt definiált, amelyet az annotálás során vizsgáltam és ellenőriztem. Később azonban az egyes hibatípusokat különböző prioritási szintekhez kötötték, majd az azonos prioritású hibatípusokat összevonták. Így korábban megfogalmazott 10 (9) hibaosztályból végül 5 független hibaosztályt definiálhattunk. A teljes annotációs folyamat azonban még az eredeti 10 hibatípussal történt, az 5 hibatípusra átalakítást egy script segítségével implementáltam.

A következő képeken a paneleken előforduló hibákat szemléltetem a prioritás színekódjaival (melyet a 2.1 táblázatban ismertetek):



2.3. ábra. Napelem panelen lehetséges hibafajtái (a prioritási szinteket jelölő színekkel: piros - magas, kék - közepes, zöld - alacsony)

1. *Zártatos string (Short circuit string)*: Az egész string (egymás feletti cellák alkotta oszlop(ok)) „sötét”, mivel egy-, vagy több cella zártatos lett. Ez tekinthető a legkritikusabb hibának, mivel a napelem nagy része nem termel áramot.
2. *Zárlat (Short circuit cell)*: A cellák sötétek maradnak a felvételen. Energetikai szempontból fontos a detektálásuk, hisz ezen a területen a panel nem képes a nap sugarainak befogadására és ezáltal az áramtermelésre sem.
3. *Alacsony teljesítményű cellák (Low power cell)*: Nem világítanak olyan nagy intenzitással, mint a többi cella, így nem biztosítanak akkora teljesítményt sem, ezáltal kritikus hibának számítanak.
4. *Repedés (Crack)*: Hasonló a mikrorepedéshez, de ekkor végigrepedt a cella. A levált cellarész (fekete háromszög) esetleg már nem is működik, emiatt ez a hibafajta sokkal kritikusabbnak számít, a napelem energia-termelését befolyásolhatja.
5. *Mikrorepedés (Micro crack)*: Általában "busbar" (függőleges cellákat összekötő vékony fémszál) mellett keletkező apró repedés, ami nem be-

folyasolja a cella működését számottevően. Ez a hibatípus szerepelt leg-többször az annotációk során.

6. *Szennyeződés (Contamination)*: Gyártás során valamilyen szennyeződés került a cellára/stringre. Nagyon hasonlít a folthoz, amikor egy-egy cel-lán található, de a legtöbb esetben több egymás alatti cellán kiterjedve látható a szennyeződés.
7. *Folt (Dark spot)*: Fekete folt a cellán. Okozhatja hő, sokk (erős nyomás), vagy törés. Ezen a hibafajta detektálása az egyik legnehezebb, mert nagyon hasonlít a szennyeződésre.
8. *Karcolás (Scratch)*: Egy kisebb, vízszintes repedésszerű csík egy-egy cel-lán belül. Nem a busbar-tól, hanem a cella közepétől indul, és halvá-nyabb, mint egy repedés. Ez a hibatípus kevésbé befolyásolja a napelem működését. Ha a készített kép minősége, vagy a kontrasztja nem a megfe-lelő, akkor ennek a hibának a detektálása nem lehetséges (túlexponálástól eltűnhet, míg alul-exponálástól beolvadhat a háttérbe).
9. *Ismeretlen hiba (Unknown error)*: Olyan hibák tartoznak ide, amelyeket nem lehet besorolni semelyik fenti osztályba. Mivel az annotálás során nagyon minimális számban jelent meg ez a hibafajta, így a megbízóval egyeztetve ezt a hibát elvetettük.

Számos hibatípus összevonásra került. A következő táblázatban az egyes hi-batípusok prioritási sorrendje látható az összevonást követően:

2.1. táblázat. Priorizált hibatípusok sorrendje

1. Zárlatos string		A legsúlyosabb hiba	
2. Zárlatos cella		Nagyon súlyos hiba	
3. Repedés		Súlyos hiba	
4. Micro repedés		Kevésbé súlyos hiba	
5. Egyéb hiba		Legkevésbé súlyos hiba	

Az összevonás során ($10 \rightarrow 5$) "zárlatos cella" osztály a zárlatos és alacsony teljesítményű osztályok összevonásából, míg az "egyéb hiba" osztály pedig a szennyeződés, folt és a karcolás összevonásából keletkezett. A táblázatban

található színek az energetikai szempontú prioritások fontosságát mutatják. A legfőbb cél, hogy hibás panelt semmiképpen se minősítsünk jónak. Tehát ha két hibatípus között nem tudunk dönteni, a súlyosabb hibát kell választani. Pl.: Ha nehezen, vagy nem eldönthető, hogy egy adott hiba Repedés, vagy Zárlatoscella, akkor a nagyobb prioritású, súlyosabb hibának kell minősíteni.

A napelem paneleken található hibák alapján a minősítés egy pontrendszer alapján történik, minden hiba az adott prioritásnak megfelelő mértékben határozza meg a saját pontszámát. A következő táblázat az adott hibatípusokhoz tartozó hibák pontszámát definiálja:

2.2. táblázat. Adott hibákhoz tartozó minősítési pontok

Hiba megnevezése	Hibapont
1. Zárlatos string	60
2. Zárlatos cella	20
3. Repedés	15
4. Micro repedés	5
5. Egyéb hiba	5

Egy panel maximum 100 pontot kaphat, amelyen nem található hiba. Adott panel pontértéke úgy számolható ki, hogy a maximális pontból ki kell vonni a szóban forgó panel hibáinak összetett pontértékét. Ezáltal a kevert hibatípusokkal rendelkező panelek is osztályozhatók. Illetve akkor is könnyen kategorizálható, ha csak egyetlen hibatípus jelenik meg a panelen, pl. egy zárlatos stringgel rendelkező panelérték 40 pont (minősítés: selejt).

2.3. táblázat. Példák a minősítési pontozási rendszerre

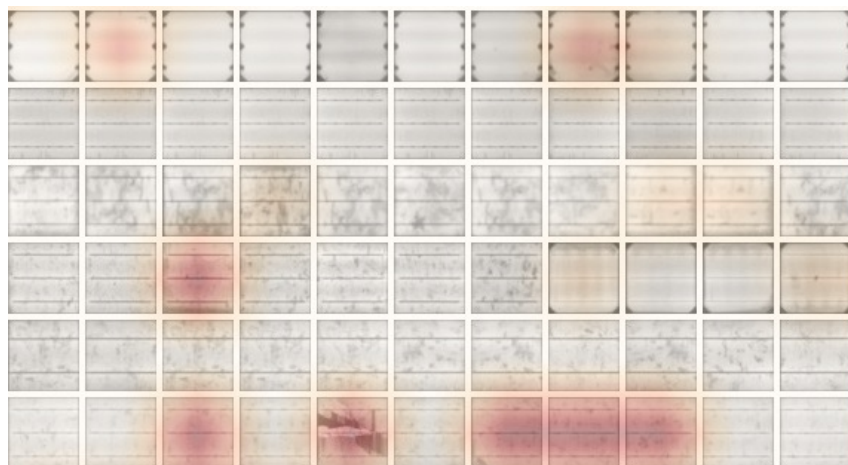
Példák a pontozásra			
Hibák	Hibapont összesen	pont (max-hibapont)	Minősítés
1 zárlatos stringel	60	40	selejt
1 zárlatos cella	20	80	megfelelő
1 zárlatos cella + 1 db repedés	35	65	megfelelő
1 zárlatos cella + 1 db repedés + 2 db mikro repedés	45	55	selejt
1 db repedés + 4 db mikro repedés + 1 db egyéb hiba	40	60	megfelelő
5db mikro repedés + 2 db egyéb hiba	35	65	megfelelő
1db mikro repedés + 1 db egyéb hiba	10	90	kiváló

3. fejezet

Adatbázis

Napjainkban a napelem panelek minősítésének automatizálása nagy kihívást jelent, ezért egyre több kutató-fejlesztő csoport kezdett el a témával mélyebben foglalkozni [6]. A kutatások során a napelem panelek változó szerkezete miatt (anyagi és szerkezeti) több adatbázist is publikáltak, amellyel az annotálás és a mintagyűjtés folyamatát is segítették egyben, ezáltal az algoritmizálás folyamatára kellett csak koncentrálni. A különböző irodalmi hivatkozásokban használt szabványos adatbázisokból a két legfontosabbat fogom bemutatni (ELPV, TecNALIAPR), melyeket a saját ELDDS adatbázisunkkal is összehasonlítottam.

3.1. ELPV



3.1. ábra. ELPV napelem panel kinézete

Buerhop-Lutz 2018-ban publikálta az ELPV adatbázist, melyhez egy GitHub repository-t [7] is létrehozott az EL-képek feldolgozására. Így publikusan,

bárki által elérhető adatbázis tudja kiszolgálni a kutatócsoportokat, megszüadítva őket a hosszadalmas annotációs folyamattól. Az adatbázis összesen 2624 mintaképet tartalmaz: 1116 hibamentes és 1508 hibás napelem cellát. Két hibaosztály kiegyenlítettlen, melyen az adatbázis augmentálásával lehet javítani (ELPV Augmented). A hibák halmaza a mikrorepedésektől kezdve a törésekig terjed, mind kisebb-nagyobb méretben található. Minden kép egy foto-voltaikus kicsi napelem cellát (PV) tartalmaz, ezért a képek mérete 300×300 -as 8-bites szürkeárnyalatos. Az adatbázis eredetileg 42 napelem panelből lett kiválogatva, majd ezek a képek normalizálva lettek a perspektívához és a mérethez viszonyítva. Az adatbázis monokristályos és polikristályos napelemeket is tartalmaz. A probléma megoldására többféle osztályozási módszert alkalmaztak:

- Az osztályozási problémát egyfajta bináris problémának tekintették, így két osztályt különböztettek meg: hibás és hibátlan panelek kategóriáját. Emiatt a probléma kevésbé komplex, ez biztosította a jobb eredményeket.
- Az osztályozási problémát valószínűségi értékekhez köthető csoportokkal valószínűzték meg: 0.0/0.33/0.66/1.0 kategóriák azt jelölik, hogy az adott cella milyen mértékben volt hibás.
- Az osztályozási problémát az általunk is azonosított kategóriánkhöz tartozó csoportokkal valószínűzték meg (YOLO-PV).

3.2. TecniaPR

Ennek az adatbázisnak a felvételeit a Tecnia kutatási központban a [8] Endeas QuickSun 600Lab napelem szimulátorával készítették egy integrált EL rendszeren. A keretrendszert PROMISE projektnek nevezték el (KK2019/00088), mely a napelem panelek vizsgálatával foglalkozó feladatokat foglalja össze. A rendszerben a képek készítésére 200 um felbontás érhető el, 4 IR CCD kamera (8.3 MP) segítségével. A 4 kamera segítségével a napelem panel 4 különböző szeletét vették fel, melyeket a nagyobb felbontás miatt később összeillesztettek. Így a képek nem egy-egy cellát, hanem a teljes panelt reprezentálják, emiatt a képek mérete is jóval nagyobb. A hibák jórészt külső környezeti hatásokból származnak, és nem a panel belső tulajdonságaiból. A

képek mérete 7942×4096 -os, 8-bites szürkeárnyaltos. A cellák száma képenként 60 és 120 közötti. Összesen 5592 cellát tartalmaz az augmentált adatbázis: 3 gyűjtősín (2592), mint 5 gyűjtősínnel (1800) és hosszított (1200). Két osztályozási módot használva 4885 cella volt hibátlan, 707 pedig hibás. Itt a hibák nagy részét a mikrorepedések, a törések és a zárlatos cellák alkotják.

3.3. ELDDS - saját adatbázis

A korábbiakban ismertetett két publikus adatbázis (ELPV, TechnaliaPR) sajnos nem volt alkalmas az általunk használt nanopel panelek vizsgálatára, annak eltérő anyagi és szerkezeti tulajdonsága miatt. Ezért a napelem gyártótól (CorecommSI Kft.) kapott EL felvételekből egy saját adatbázist hoztam létre. A neurális hálózatok felügyelt tanítása során rendkívüli jelentőséggel bír a tanító-validáló-tesztelő adathalmaz megfelelő és pontos kialakítása.



3.2. ábra. ELDDS - egy napelem panelről a gyártói EL tesztelő berendezésben készült felvétel

Kezdetben a feladatom az volt, hogy a korábbi adatokat és a velük elvégzett tanításokat és teszteléseket tüzetesebben megvizsgáljam. A folyamat során az annotációs fájlokkal összevetve sok ellentmondást és hiányosságot tapasztaltam, pl.: nagyon sok hiba nem volt megfelelően annotálva, és hiányos annotálásokat is találtam.

Ezt az adatbázist "*ELDDS I. - EL Defects Data Set I.*" névre kereszteltük, mely 500 db tanító képet és 45/87 tesztelő képet tartalmazott. Az adathalmazt a gyártó által kijelölt két annotátor egymástól függetlenül dolgozta fel. A vizsgálatunk során sajnos az tapasztaltuk, hogy az adatbázist nem konzisztens módon sikerült annotálniuk. Az annotáláson kívül a napelem panel felvételeken különböző problémák jelentkeztek, amelyek befolyásolták a neurális hálózat tanítási pontosságát:

- Változó intenzitásúak a felvételek.
- Nagy felbontású (ált. 12 MPx), de éleetlen képek.
- Piszkos-foltos régiók a plexin, amelyre a napelem paneleket fektették (nem panel hiba!).
- Bizonyos hibafajtákból (pl. „scratch”) kevés van.

Az alábbi táblázat az annotálások közötti különbségeket szemlélteti:

3.1. táblázat. Annotációk közötti különbség (ugyanazon adathalmazon)

	Annotator 1	Annotator 2
Micro crack	1240	711
Crack	374	235
Short circuit cell	311	252
Contamination	1	267
Dark spot	28	75
Unknown error	31	2

Emiatt kezdetben a hibák azonosítása is sok nehézséget okozott a számomra. A hibás annotálások kiszűrését és többszörös iteratív ellenőrzését is elvégeztem az összes képre. Mivel az adathalmaz eloszlását tekintve bizonyos hibafajtákból (pl.: karcolás) csak kevés állt a rendelkezésre, ezért új képekre volt szükségem, amelyeket a napelem paneleket gyártó cég biztosított. Ezt 2021. augusztusában kaptuk, 12 GBytos csomagban, és kb. 3000 EL képet tartalmazott. Elsőként ezen az adatbázison is szűrési és tisztítási feladatokat végeztem: egyrészt mivel a felvételek közel fele szerepelt a korábbi, már felhasznált annotációkban, másrészt olyan képeket is tartalmazott, amelyeken csak egyetlen cella (nem a teljes panel vagy string) volt található. Voltak

olyan képek is, amelyeknél a napelem cellák szerkezete eltért a korábbiakhoz képest: a "busbar" mentén töréshez hasonlítható háromszögek voltak, de vizsgálataim során a tanítás és tesztelés folyamatainál ez a szerkezeti különbség nem befolyásolta az eredményeket. A deduplikáció ellenőrzése során egyrészt a hash (hasító) függvényt alkalmaztam, másrészt név alapú összehasonlítást is végeztem a képek szelektálására. Ekkor hoztam létre az ELDDS II. nevű adatbázist, mely további 450 db tanító-, és 65 db tesztképet tartalmazott, de ezeket a képeket a korábbi ELDDS I. adatbázis nem tartalmazta (metszetük üres). Az augusztusi képekből még kb. 660 db kép vár kézi annotációra, ami majd az ELDDS III. nevű adatbázist alkotja. A különböző neurális hálózatok tanítása az ELDDS I.-II. adatbázisok alkalmazásával történt, amely összesen így *1240 db tanító, és 140 db tesztelő* képet tartalmaz. A tapasztalataim alapján a tanító képek minősége fontos szerepet játszik a nagyobb pontosság elérése érdekében.

4. fejezet

Képek előfeldolgozása

A tanítási és tesztelési eredmények javításának egyik lehetséges módszere az annotált képek előfeldolgozása. Erre különféle módszerek választhatóak, a napelenk panelek szempontjából a legfontosabb változtatás a képek kontrasztjának és fényességének módosítása bizonyos képfeldolgozási műveletekkel:

4.1. Hisztogram transzformációk

A hisztogramszéthúzás során a képek sötétebb részeit világossá alakítom, a nagyon világos részeket pedig sötétté a jobb láthatóság miatt. Ezáltal több információt kaphatunk meg egy képről. A vizsgált adathalmazon az derült ki, hogy a világos képekre a négyzetes hisztogram széthúzás, míg a sötéteknél a gyökös széthúzás volt eredményesebb. Számítási képlete a következő:

$$y = \frac{255}{max - min}(x - min) \quad (4.1)$$

A hisztogramkiegyenlítés során a kép kontrasztja növelhető meg. Az eljárás során a bemeneti kép színeit egy adott darabszámú szürkeségi fokozatba csoportosítjuk, így jobban érzékelhetőek lesznek az élek és a vizsgált alakzatok. A darabszámmal lehet szabályozni a kontraszt mértékét: ha kevés, akkor a képen az alakzatok kivehetősége csökkenthet, ha sok, akkor a kontraszt növelése sikertelen lesz. Tapasztalatom alapján 4 db esetén jobb a kontraszt, de a képen lévő alakzatok minősége csökken, míg kb. 16 db esetében elérhetünk megfelelő kontraszt viszonyt.

4.2. Wallis-szűrő

A Wallis szűrőt [9] arra használjuk, hogy a kép különböző részein lokálisan javítsuk a kontrasztot. Ha egy szűrkeskálás képen nagyon világos, ill. nagyon sötét területek is vannak, a teljes kép fényerejének átállításával nem minden terület lesz megfelelően látható. A szűrő képlete a következő:

Wallis szűrő képlet

$$W(i, j) = [F(i, j) - M(i, j)] \frac{A_{max} \times S_d}{S_d + A_{max} \times S(i, j)} + [r \times M_d + (1 - r) \times M(i, j)] \quad (4.2)$$

Ahol i, j a képpont koordinátái, W a szűrő kimenete, F az eredeti kép, M a kép simított változata (átlagoló szűrő/medián szűrő/gauss szűrő), míg S a képpontok világosság-értékeinek szórása; a felhasználó által megadott paraméterek:

Lokális átlag meghatározása

$$M(i, j) = \frac{1}{W^2} \sum_{m=i-w}^{i+w} \sum_{n=j-w}^{j+w} F(m, n), \text{ ahol } W = 2w + 1 \quad (4.3)$$

Szórásnégyzet

$$S^2(i, j) = \frac{1}{W^2} \sum_{m=i-w}^{i+w} \sum_{n=j-w}^{j+w} (F(m, n) - M(i, j))^2, \text{ ahol } W = 2w + 1 \quad (4.4)$$

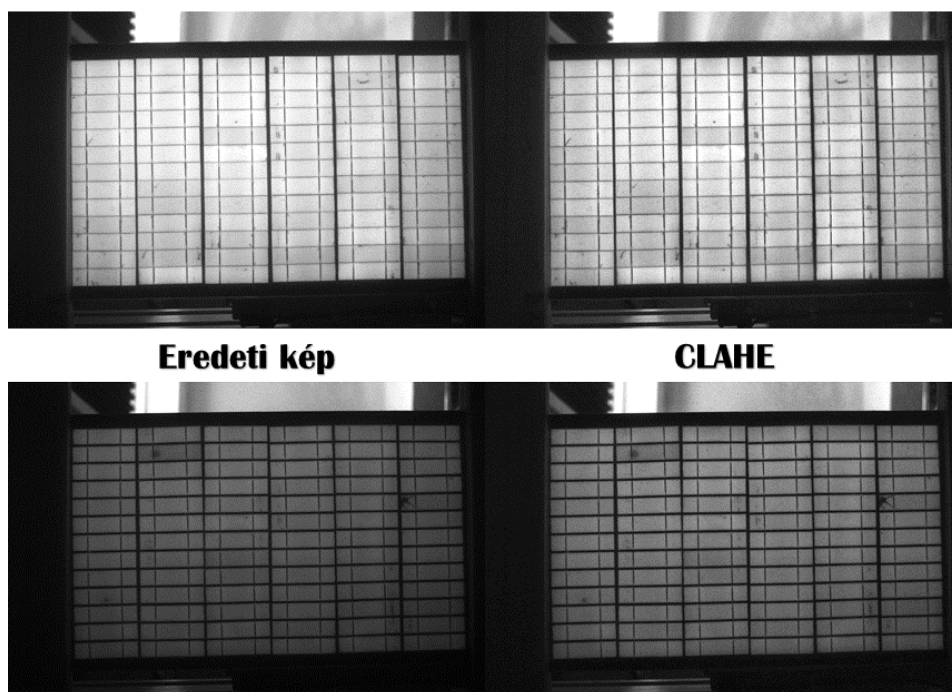
M_d az elvárt világosság, S_d az elvárt kontraszt, A_{max} és r a beavatkozás mértékei. Szintén paraméter a w , azaz a szűrőablak sugara. Alap beállítások az ideális kontraszt elérésére: $w : 8$, Elvárt kontraszt: 100; Elvárt világosság: 128, Beavatkozás mértékei: $r = 0.25f$, $A_{max} : 2.5f$. Ezen változók értékeinek megválasztásától is függhet a lokális kontraszt kialakításának minősége.



4.1. ábra. Wallis szűrő hatása

Az ábrán a Léna kép kontrasztjavítása látható. Az első képen egy zajos változata található az eredeti Léna képnek, ez jól bemutatja a Wallis szűrő egyik jelentős hatását: a kép kontrasztja mellett egy simítást is elvégez ez a szűrő. A harmadik-negyedik kép a szűrő főbb funkcióját szemlélteti: a kontraszt javítást.

4.3. CLAHE



4.2. ábra. CLAHE hatása

A Kontraszt Korlátozott Adaptív Hisztogram-kiegyenlítés, vagy röviden CLAHE (Contrast Limited Adaptive Histogram Equalization) [10] a modern képfeldolgozásban előszeretettel használt algoritmus, amely egy adott kép globális kontrasztját felhasználva erősíti a kép lokális kontrasztját. Ez a módszer számos területen bizonyult nagyon hasznosnak, például vízalatti képeknél, forgalomirányításnál, csillagászati-, és orvosi képalkotásnál. A CLAHE használható HDR (High Dynamic Range) képek megjelenítésének tónuslekepezési műveletében is.

Az algoritmus azért született meg, mert sok esetben a globális hisztogram kiegyenlítés nem javítja a kép kontrasztját, ebből kifolyólag a képek túl-, vagy alulexponáltak maradtak. Ez azért következhetett be, mert a hisztogramkiegyenlítés nem korlátozódik egy adott régióra. A CLAHE során a kép alapértelmezetten 8×8 -as kis „csempékre” (blokkokra) van osztva, amelyeken külön-külön hisztogramkiegyenlítés történik. Ha nincs zaj, akkor a hisztogram erre a kis régióra korlátozódik, ha van, akkor az felerősödik. Ezt a problémát küszöböli ki a kontrasztkorlát. Ha a hisztogram a megadott kontrasztkorlát fölé emelkedik, akkor a képpontokat levágja és egyenletesen elosztja a többi ún. bin ("vödör") között, mielőtt a hisztogramkiegyenlítést elvégezné. A kiegyenlítés után bináris interpolációt alkalmazunk a blokkok szélén lévő artifactok eltávolítására. A színes képek esetében a módszer csak akkor használható, ha a kép *CIELab*-színtérré van transzformálva, ezzel elkerülve a nem kívánt színárnyalat-, és telítettségváltozást.

A CLAHE-algoritmus a következő lépéseket használja a módszer megvalósításához [11]:

1. A kép minden pixelére meghatározza a szomszédsági mátrix hisztogramját.
2. Minden hisztogramnál egy küszöbértéknek megfelelően vágást hajt végre: az így kapott hisztogramot egyenlően elosztja a többi hisztogram "bin"-jei között.
3. CDF (Cumulative Distribution Function) kiszámítása minden pixelre, amely szerepel az átalakított hisztogramban - egy transzformációs függvényt határoz meg.

4. Végül ezt a transzformációs függvényt alkalmazva minden pixelre megkapjuk a hisztogram-kiegyenlített képet.

A CLAHE algoritmus pszeudo kódja [12] a következő:

Input: Original Image I ;

1. Resizing I to $M \times M$; Decompose $I \rightarrow (n)$ tiles; $(n) \leftarrow \frac{M \times M}{m \times m}$;
2. $H_n \leftarrow$ histogram (n) ; // histogram of a $m \times m$ tile;
3. Clip limit: $C_L \leftarrow N_{cl} \times N_{avg}$;
 // $N_{avg} \leftarrow \frac{N_x \times N_y}{N_{gray}}$;
 // $N_{gray} \rightarrow$ number of gray levels in the tile;
 // $N_x, N_y \rightarrow$ number of pixels in the x, y dimensions of a tile;
 // $N_{cl} \leftarrow 0.002$ // normalized contrast limit;
4. Clipping of H_n using C_L ;
 // For gray levels greater than C_L ; let $N_{\sum_{cl}}$ pixels are clipped;
5. $\frac{N_{\sum_{cl}}}{N_{gray}} \rightarrow N_{cp}$ pixels $\rightarrow$ distribution over the remaining pixels;
 // contrast limited histogram of each tile after pixel distribution;
6. CLAHE $(n) \leftarrow$ Equalization of contrast limited tile histogram using (1);
7. $I_c \leftarrow$ bilinear interpolation of CLAHE processed n tiles;
 // combining neighborhood tiles

Output: CLAHE processed image I_c ;

5. fejezet

Annotálás és detektálás

5.1. Mérési környezet

A napelem panelek hibadetektálásánál az elsődleges cél az volt, hogy egy olyan mérőkörnyezetet tudjunk kialakítani, amely segítségével akár egyetemi körülmények között is tesztelni tudunk. A vizsgálat során a Corecomm megbízó által biztosított ún. „Domino” napelem panelek analízisével szeretnénk volna az automatizált tesztelés - azaz a felvétel készítés, detektálás, felismerés - folyamatát kialakítani és tesztelni.

A képek készítéséhez mindenképpen szükséges egy speciális kamera, amely Elektro-Lumineszcens felvételekre képes, vagy Near InfraRed tartományban üzemelő kell legyen. Az "elektrolumineszcencia" alapú vizsgálati módszer alapgondolata, hogy elektronikai szempontból a napelem lényegében egy nagyfelületű fényérzékeny dióda. Ha képes villamos energiát előállítani a fény besugárzásából, akkor - a működés megfordításával és az áram bekapcsolásával - fényt fog kibocsátani. Azoknak a celláknak, melyek így módon az egész felületükön képesek fényt kibocsátani, az energiatermeléses állapotukban is jól kell, hogy működjenek (éppen ezért sok gyártó ebben az esetben is állapotellenőrzést végez). A napelem által kibocsátott fény spektrális tartománya főként a rövidhullámú infravörös (NIR - Near InfraRed) tartomány. Emiatt akár rövidhullámú hőleképező eszközök alkalmazásával is vizsgálhatók. Valójában a rövidhullámú termográfiai rendszereket a gyártás során általában napelem cellák ellenőrzésére használják, ami gyors gyártósori kiértékelést igényel. A foto-detektor rendszer súlya és mérete mellett ezek magas költsége is említésre méltó. A kamerákban lévő CMOS szenzorok kellően széles spektrális tartománnyal rendelkeznek, így egy beépített infraszűrő eltávolításával és egy nagy

fényerejű, rövidhullámú sugárzás továbbítására képes lencse használatával érzékelni tudjuk a napelemek emisszióját. A méréshez természetesen nagyon hosszú, akár néhány másodpercnyi záridő szükséges, az eredmény pedig nem kalibrálható. A közeli infravörös (NIR) kamera az elektromágneses spektrum infravörös sávjának egy részhalmazát használja fel a képkészítésre, amely a 700 nm és 1400 nm közötti hullámhossz tartományt fedi le. Éppen emiatt sokkal tisztább részleteket is tudunk ezzel a kamera típussal érzékelni, mivel a hullámhossz kívül esik az emberi szem számára látható spektrális tartományon. A NIR kamerával készült képek azért ideálisak számunkra, mert a képkészítés során eltávolítja a színek hullámhosszait, ami a képek szürkeskálás kinézetét eredményezik. Mivel ezek a kamerák érzékenyek a fényre, így a képek készítését sötét helyen kell elvégezni, mert a fény ronthat a napelem panel képének minőségén, ezáltal a napelemeken fellelhető halvány hibafajta (pl.: karc, vagy szennyeződés) nehezen kivehetővé válik, befolyásolva a neurális hálózat tanítási és tesztelési folyamatát. [13]



5.1. ábra. Kialakított mérési környezet az egyetemen

Az egyetemen készített kameraképeket először egy adattároló eszközre mentjük, amely esetünkben a detektálási folyamatban is használt nagyteljesítményű

munkaállomás volt. A képfelvételek során a Basler NIR speciális kameráját használtuk (acA2040-25gmNIR [14]) maximális felbontása 4.2 megapixel, az expozíciós időt 1-2 sec között állítottuk, és kontraszt emelés végett az analóg erősítés (gain) értékét is magasabbra ($\sim 50-75$) állítottuk. Ezenkívül a napelemek tápellátását egy labortápegység biztosítja (szabványosan $\sim 47-48V$), mellyel a szükséges feszültség szintek kapcsolhatók a napelem panelek hátoldalán lévő kétpólusú kivezetésekre.



5.2. ábra. Felvételek készítésénél használt Basler kamera - acA2040-25gmNIR

A kismértékű szabályozásra azért van szükség, mert egyes hibafajták a panel állapotától függően a szabványos feszültségen alig voltak láthatóak. Minél nagyobb a tápfeszültség, annál nagyobb intenzitású képet kapunk a napelem panel EL kibocsátásától függően (de nyilvánvalóan a szabványos tápfeszültség fölé csak max 1-2 V-os felső küszöb eléréséig mehettünk). Azt tapasztaltam, hogy minél kisebb a tápfeszültség (küszöbfeszültség alatt), annál jobban kivehetők a "vonalszerű" sötétebb hibafajták pl.: mikrorepedések, vagy repedések.

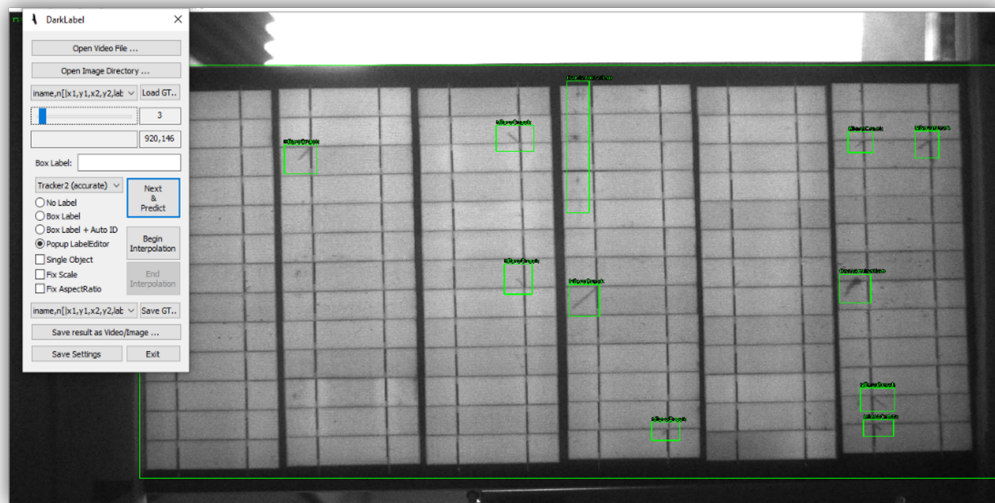
Mivel a neurális hálózatok tesztelését akár CPU-n is el lehetne végezni, ezért az általam feldolgozott adatbázisokból készített tanítási modelleket a számítógépen tárolva könnyedén elérhetjük, a tesztelés folyamata rövid időn belül végrehajtható. A tesztelést több neurális hálózaton is elvégeztem, melyet egy későbbi fejezetben mutatok be részletesen. Céloom a legjobban "teljesítő" neurális hálózat kialakítása, paramétereinek módosítása és tanítása a képi adatbázison. A tesztelés és kiértékelés során az általam tanított és hangolt neurális hálózattal eredményül kapjuk a detektált és felismert objektumokat (hibákat). Végül pedig az általam készített „Solaris” névre keresztelt Python nyelvben íródott GUI alkalmazásban a minősítési adatok kinyerhetők a panelekről.

5.2. Annotáló eszközök és a formátum átalakító alkalmazás

A korábbi adatbázison (ELDDS I.) végzett annotálás során egy távoli szerveren futó online annotáló eszközt használtak, ez volt az *AnnotationTool_3*, melyben a befoglaló doboz (bounding box) alapú kézi bejelölést egy struktúrált .json fájlban tárolta. Minden képhez tartozott egy "BoundingBoxes" rész, amely a befoglaló dobozokat tartalmazta: egy sorszám index, szélesség, magasság, és a középpontok a koordinátái. Viszont ez program a feladatom kezdetén már nem működött (a szerver már nem volt elérhető), így egy másik annotáló programot, a DarkLabelt [15] használtam. Elsőként az annotálás során a régi annotációkat kellett átalakítanom az általam használt DarkLabel formátumra. Erre a feladatra készítettem el a „Solaris” néven létrehozott formátum átalakító grafikus alkalmazásomat Python nyelven (mivel a neurális hálózatok keretrendszere is ezt a nyelvet támogatja).

5.2.1. DarkLabel

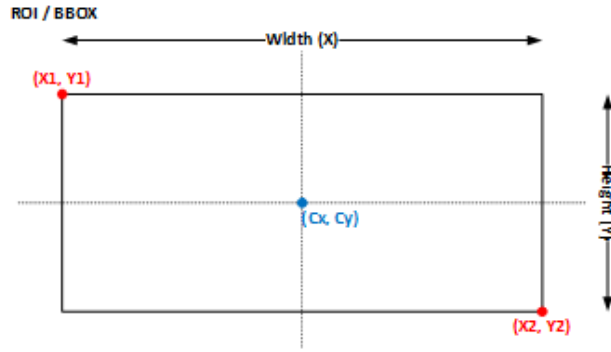
A kézi annotálás során a DarkLabel szoftverben a következő módon történik az egyes befoglaló dobozok kialakítása:



5.3. ábra. DarkLabel annotációs program

Nagyon fontos, hogy a képen megadott *"iname,n[/x1, y1, x2, y2, label]"*

formátumban legyen betöltve (LOAD GT..) és kimentve a kézzel annotált adatahalmaz (SAVE GT...), ahol a paraméterek a következők: iname: file neve kiterjesztéssel; n az adott képen annotált fizikai objektumok számát jelöl; / jelenti, hogy minden befoglaló doboz új sorba kerül; $x1, y1$: bal felső sarok koordinátái; $x2, y2$: jobb alsó sarok koordinátái; label: a címke-azonosítója a hibának. Ezt a formátumot használva az AnnotationTool_3 json formátumban leírt befoglaló négyzeteket a következőképpen konvertáltam: a json struktúra bounding boxes mezőjében megadott *Height-Width*, *CenterX*, *CenterY* értékek arányokat definiálnak, melyek az eredeti képméret pixelben kifejezett értékéhez viszonyítva adottak. A kép méretéből kiszámítva a következő módon történik az átalakítás:



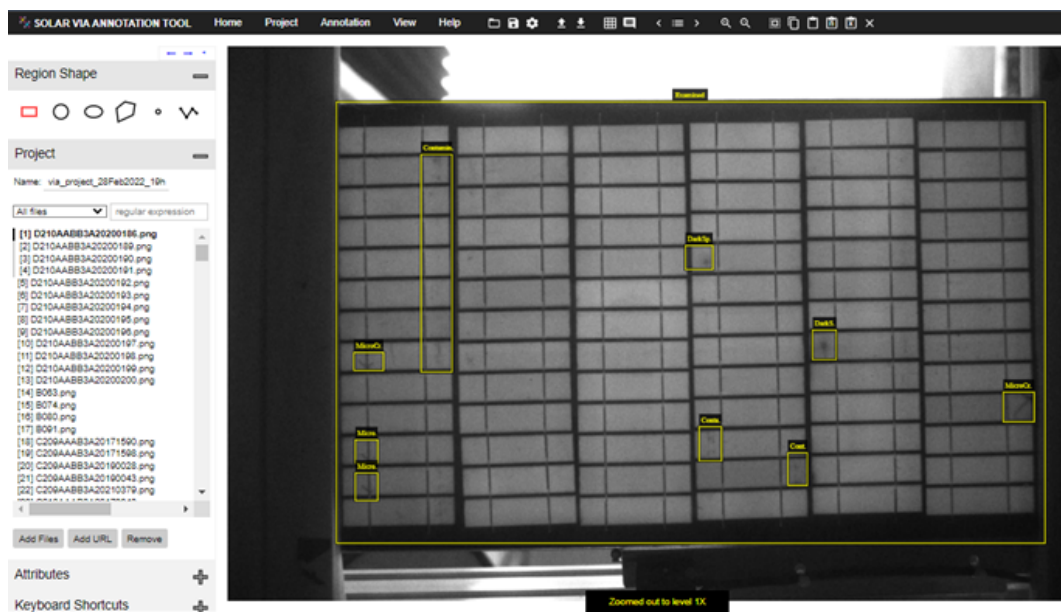
5.4. ábra. Befoglaló doboz struktúrája

A kép szélességét be kell szorozni az AnnotationTool_3 formátumban szereplő *Width* és a *CenterX* arányokkal, majd a kép magasságát be kell szorozni a *Height* és a *CenterY* arányokkal, így megkapjuk a négy érték pixelbe konvertált végeredményeit. Ezekből a következő képletekkel például a detekció során használt neurális hálózat (a RetinaNet) annotálási formátumban szereplő négy koordináta számítható ki ($X_1, Y_1; X_2, Y_2$):

$$\begin{aligned} X_1 &= C_x - \frac{W}{2}, Y_1 = C_y - \frac{H}{2} \\ X_2 &= C_x + \frac{W}{2}, Y_2 = C_y + \frac{H}{2} \end{aligned} \quad (5.1)$$

5.2.2. VIA annotációs program

Mivel a kézi annotációk javítása és az újak felvétele nehézkessé vált a Dark-Label-ben (a nagy képméret, és a nagymennyiségű képi adathalmaz betöltése miatt, amelyek sokszor az alkalmazás lefagyásához vezettek), ezért egy új annotáló programot készítettem, amely a VGG Image Annotator (VIA) [16] alapján készült, hasonló működéssel, kisebb változtatásokkal. A módosított alkalmazást *"Solar VIA"*-nak neveztem el.



5.5. ábra. VIA annotációs program

Az alkalmazást JavaScriptben implementáltam, így bármely böngészőben off-line módon használható. Az annotációk betöltése egy .json kiterjesztésű strukturált VIA project fájl segítségével történik, amelyet a Solaris alkalmazás használata során generáltam (a DarkLabel fájlból konvertálva). A project fájl a következőket tartalmazza: a VIA működéséhez szükséges beállítások, a képekhez tartozó befoglaló dobozok tulajdonságai (x , y , $szélesség$, $magasság$), minden doboz típusa „rect”-re van állítva. A fájl végén találhatóak meg az osztályok címkéi és az adatbázisban található képek neveinek listája.

A betöltés után a képek még nem jelennek meg, azokat külön be kell tölteni az „Add Files” gomb segítségével. Betöltés után a képernyőn megjelenik a nap-
elem panel és az annotálásból származó befoglaló dobozok (ROI) a megfelelő

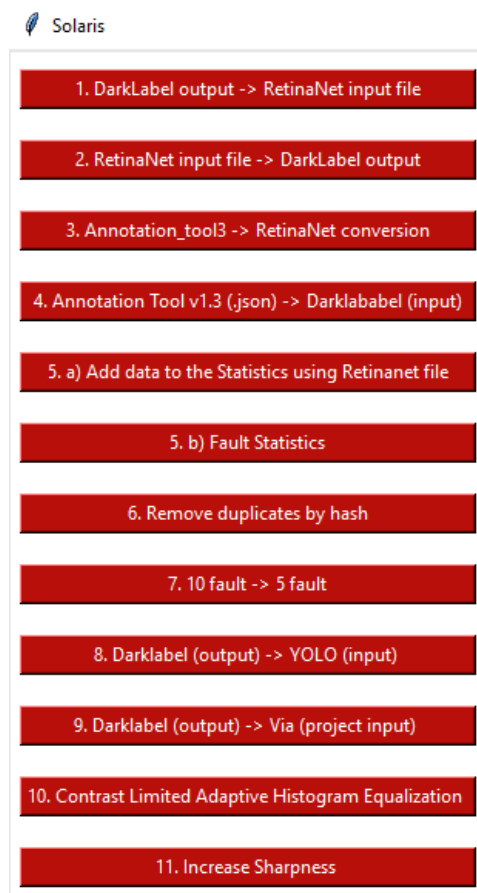
címkékkel. A címkék előre be vannak konfigurálva, amelyeket akár meg is lehet változtatni, törölni lehet, vagy éppen új címke hozzáadható a listához.

Új befoglaló dobozok rajzolására is lehetőség van az objektum körül. Az annotálás mentését a „Project -> Save” menüpontban lehet elvégezni. A mentés eredménye egy .zip fájl lesz, amelyben három fájl található:

- egy DarkLabel fájl az ellenőrzés szempontjából, és bizonyos konvertálások a Solaris alkalmazásban a DarkLabel formátumot használják;
- egy *retinanet.csv* fájl, amely a RetinaNet neurális hálózat tanítására alkalmas formátumot tartalmaz;
- és egy VIA project fájlt, melynek a betöltésével későbbi annotálási feladatok végezhetők el.

5.2.3. Solaris formátum átalakító program

Amikor a DarkLabel segítségével megkapom a befoglaló dobozok adatait, akkor azokat a különböző, vizsgált neurális hálózatok (pl. RetinaNet, YOLO) egyedi bemeneti annotációs formátumaihoz is igazítani kell, amelyet az alkalmazásom gombjaihoz társított funkcióval lehet elérni. Az alábbi ábrán az alkalmazás funkcióit jelölő gombok láthatók:



5.6. ábra. Solaris formátum átalakító program

1. DarkLabel formátumból a RetinaNet [17] [18] hálózat tanításához és teszteléséhez szükséges annotációs formátumú szöveges fájlt állít elő. A RetinaNet esetében a következő formátum támogatott: egy szöveges fájlban vannak az annotációk, minden sorban a következő található:

kép elérési útvonala, $x1$, $y1$, $x2$, $y2$, osztálynév

2. RetinaNet formátumból a DarkLabel formátumú szöveges fájlt állít elő. Ez a funkció azért kellett, mert voltak olyan annotációk, amik csak RetinaNet-es formátumban voltak eltárolva (ELDDS I. annotációi). Ez az annotációk helyességének ellenőrzése szempontjából fontos funkció.
3. AnnotationTool_3 formátumból RetinaNet tanításához és teszteléséhez szükséges annotációs formátumú szöveges fájlt állít elő. Ez a funkció csak a feladat elején, a korábbi annotációk feldolgozásánál volt hasznos.

4. AnnotationTool_3 formátumból DarkLabel formátumú szöveges fájlt állít elő.
5. A hibák statisztikájának lekérdezésére hasznos ez a funkció, az első gombbal a nyilvántartott darabszámokhoz rendelhető a hibaszám, amelyet a RetinaNet formátumú szöveges fájlból kérdezek le. A második gombbal egy diagram rajzolható ki statisztikával.
6. Az adott hash függvény segítségével kiszűri a felhasználó által megadott könyvtárból az ismétlődő képeket (deduplikálás).
7. Ez a hasznos funkció a napelemgyártó hibaösszevonásának kezelését támogatja. Lehetőséget biztosít a 10-, illetve 5 vizsgált hibafajta (osztály) közötti konverzióra.
8. A YOLO [19] [20] neurális hálózat implementálásával új annotációs formátumra volt szükség. A YOLO esetén minden egyes képhez tartozik egy a képpel azonos szöveges fájl, aminek minden sora a következőket tartalmazza: *osztály_id x_középpont y_középpont szélesség magasság*, viszont ezek az értékek 0 és 1 között lehetnek a kép teljes szélességéhez és magasságához arányosan viszonyítva (az AnnotationTool_3-hoz hasonlóan). A könyvtárat megadva a program elvégzi a képekhez való szöveges fájl rendelését.
9. A későbbiekben vizsgált neurális hálózatokhoz készült funkció, ami a szabványos COCO annotációs formátumot támogatja.
10. CLAHE (Contrast Limited Adaptive Histogram Equalization) használata - képek hisztogramjának kiegyenlítésére szolgáló funkció, a jobb tanítási és tesztelési eredmények elérése érdekében.
11. Kép élességének javítását lehetővé tevő funkció. A következő K konvolúciós kernellel valósítom meg a kép élességén és kontrasztján történő javítást:

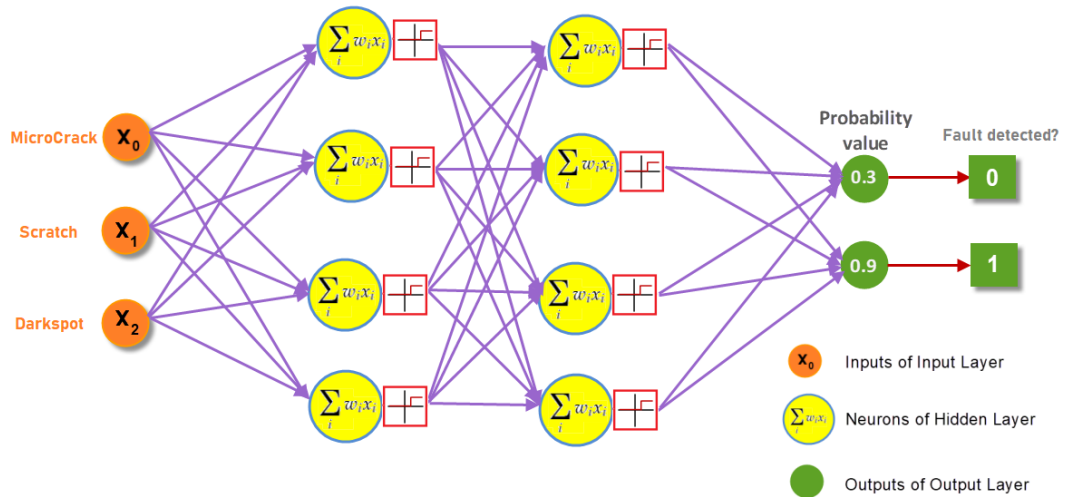
$$K = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad (5.2)$$

6. fejezet

Neurális hálózatok

6.1. Neurális hálózatok általánosan

A neurális hálózatok jól strukturált rétegekben oldják meg a tanítási és tesztelési folyamatokat [21]. Ezek a rétegek a következők:



6.1. ábra. Neurális hálózatok felépítése

- Bementi réteg: módosíthatatlanul továbbítja a bemenetként átadott adatot a hálózat többi rétegének. Egy neurális hálózatnak több bemeneti rétege is lehet és elágazásokat is tartalmazhat. A neuronok számát a tanítás által bevitt adatok dimenzionalitása és alakja determinálja.
- Rejtett rétegek: a bemeneti- és a kimeneti rétegek között helyezkednek el, az adat transzformációja, kódolása, valamint absztrakciók, köztes rep-

rezentációk megalkotása a feladata. Számuk, típusuk, valamint a neuronok száma a hálózatban megváltoztató. Általam használt YOLOv5 hálózatnak például több, mint 300 rejtett rétege van!

- **Kimeneti réteg:** a kimeneti függvényt, valamint a kimeneti neuronok számát az adott probléma jellege határozza meg. Osztályozás esetén jellemző módon annyi kimeneti neuron van, ahány kategória áll rendelkezésre, a kimeneti függvény pedig az adott osztályba tartozás valószínűségét hivatott reprezentálni a csoportok között.
- **Aktivációs függvények:** a neuronrétegek között sokféle aktivációs függvényt alkalmazhatunk. A bemeneti mátrix(képek) összes elemére egyével alkalmazzuk a feldolgozás során.
- **Veszteségfüggvények:** A "gradiens-csökkenés" módszerének megoldása esetén választanunk kell egy olyan függvényt, mely egy deriválható és egy objektív számértékként akkumulálja a hálózat hibáját a számított kimenet, továbbá az elvárt kimenet ismeretében.

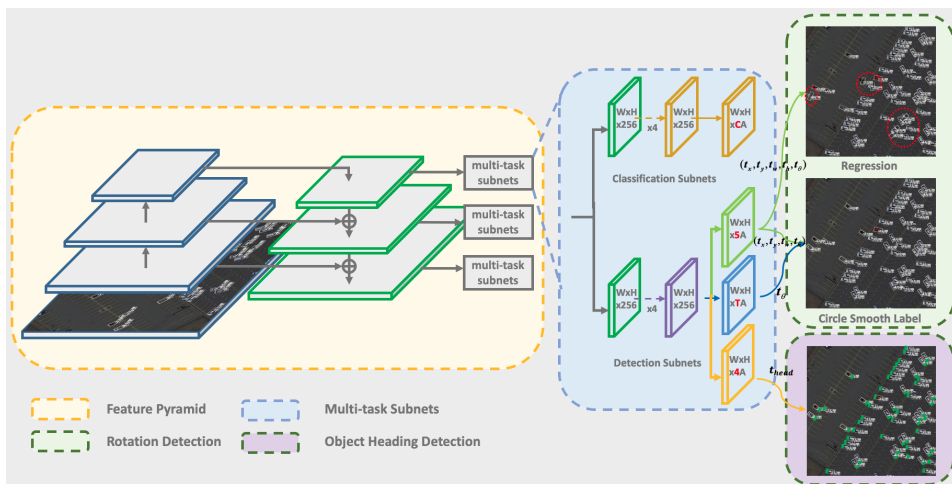
6.1.1. Egy-, és kétlépcsős detektor hálózatok

Egy neurális hálózat a tanítási folyamat során történő detektálást egy-, vagy két lépésben tudja megoldani [22]. Így kétfajta hálózati típust tudunk megkülönböztetni, melyek a következők:

1. Az *egy-lépcsős detektorok* előrecsatolt, teljes mértékben konvolúciós hálózatot tartalmaznak, amely egyszerre határozza meg a határoló dobozokat, valamint az objektum osztályozást. A Single Shot MultiBox Detector (SSD), valamint a YOLO (You Only Look Once) voltak az elsők között, akik előfeldolgozás nélküli egységes architektúrát javasoltak. Azonban a képek előtér-háttér kiegyensúlyozatlanság megakadályozta, hogy magas pontosságot érjenek el. Elsőként a RetinaNet próbálta kezelni ezt a nehézséget a piramishálózat használatával és veszteségfüggvény módosításával - csökkenti a pontosságot az egyszerű mintákon, hogy a nehezen detektálható tárgyakra összpontosíthasson. A YOLO detektorok egy másik "anchor" - "horgony" alapú feldolgozást használnak, amely

felosztja a képet régiókra, valamint előre jelzi az egyes régiókhoz tartozó határolódobozokat, ezzel gyorsabb következtetési arányt érnek el, mint a RetinaNet, bár kevésbé pontosak. Újabban egy másik népszerű megközelítés az anchor-mentes megvalósítás terjedt el. Például az FCOS célja, hogy egyszerűsítse az objektumészlelési nehézséget az objektumok középpontját használva. Az FCOS képes felülmúlni a RetinaNet pontosságát, miközben egyszerűbb és flexibilisebb, mert nincs szükség anchor-ok pontos beállítására. Egyéb metódusok, mint például a CornerNet, vagy az ExtremeNet első lépésben több kulcspontot keresnek, majd létrehozzák a határolódobozokat az objektumok észleléséhez (utófeldolgozás sebességének kárára).

2. *A két-lépcsős detektorok* a detektálási és osztályozási folyamatokat külön-külön lépésben oldják meg. Ezek a modellek első lépésben néhány régiót, mint jelöltet (candidate RoI) javasolnak. A második lépésben a javaslatok osztályozása, majd utána azok helyzetének finomítása történik meg. Az R-CNN hálózat osztályozást és a határolódoboz alapú regressziót használ tanítás során. Később a Faster-RCNN-val megjelent a konvolúciós gerinchálózat (backbone) használata, mint például a VGG vagy a ResNet, amely globális tulajdonság-térképeket (feature map) generál. Ezek a hálózatok megosztott konvolúciós-térképeket alakítanak ki a regionális ajánlati hálózat (RPN) és az észlelési hálózat között, amely csökkenti a javaslatok generálásának költségét. A detektálás pontosságának javítására megjelent a piramishálózat (FPN) - a különböző szintek közötti ROI jellemzők kivágása történik, amely a méretarány értékétől függ. A Cascade R-CNN több detektor sorba kapcsolását jelentette, melyek a megnövelt crossover union (IoU) küszöböt használják a tanításra, ezzel megbízhatóbb megfigyeléseket sikerül elérni, de növeli a szükséges számítási költséget.



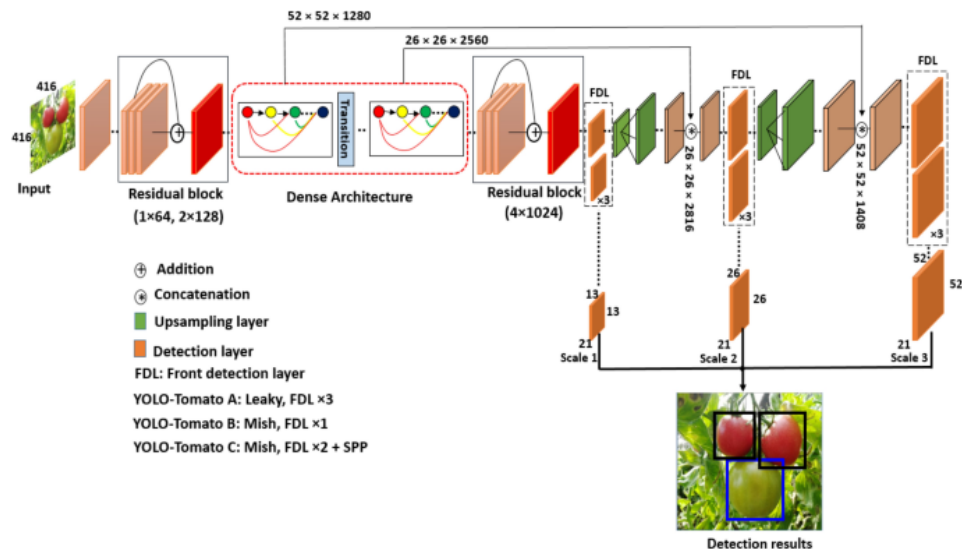
A RetinaNet a következő részhálózatokból épül fel:

1. A gerinchálózat kimenetén konvolúciós objektumosztályozás történik
2. Konvolúciós befoglaló doboz segítségével regresszió történik.

A RetinaNet verziók közötti különbségek: ResNet-152, ScaleNet, mint újabb, hatékonyabb gerinchálózatok megjelenése. Jellemzők egyesítése (Feature fusion) a kisebb méretű objektumok detektálásánál lehet hasznos.

6.2.2. YOLOv3 és YOLOv5

A *YOLO* (*You Only Look Once*) [19] [20] egy-lépcsős detektor az újabb neurális hálózatok közé tartozik, főképpen a valós idejű alkalmazásokban használják. A hálózat alapötlete az volt, hogy a képet több cellára osztja egy rács segítségével, ezzel a detektálás kisebb méretekben történik – ezzel gyorsítva a tanítás folyamatát. A YOLO-nál az objektum detektálásra regressziós probléma megoldását jelenti: minden cellában több, előre definiált anchor doboz van, amely lehetővé teszi a befoglaló dobozok egyidejű kiszámítását, a konfidencia értékének és az osztály valószínűségek meghatározását. Az előzetes anchor dobozok dimenzióinak meghatározása pedig a K-mean klaszterezés segítségével történik, a távolság metrikán alapuló tanító (ground truth) dobozokkal.



6.3. ábra. YOLO hálózati topológiája

A YOLO verziók sajátosságai:

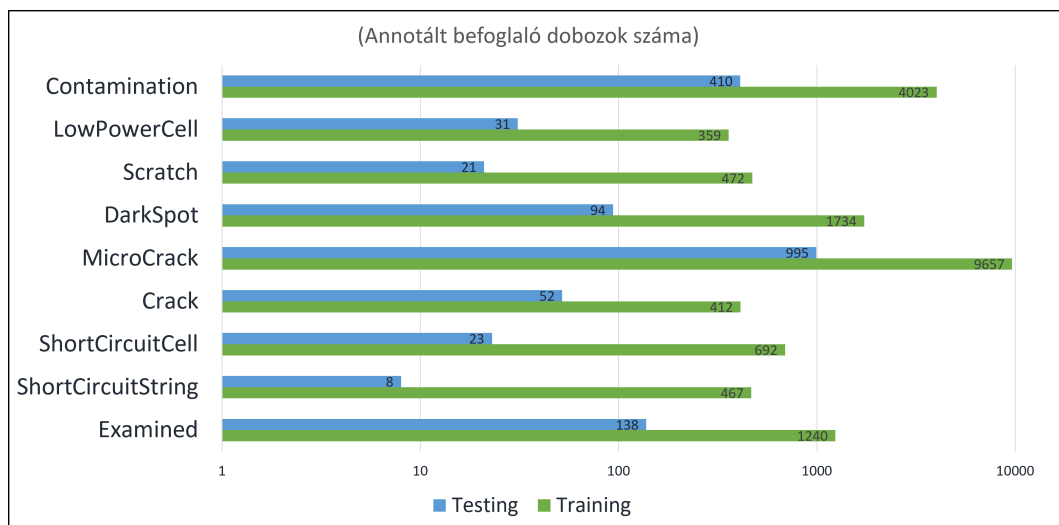
- YOLOv3: többléptékű észlelés, javított gerinchálózat, módosított veszteségfüggvény
- YOLOv5: továbbfejlesztett modell a sebesség, pontosság, méret szempontjából. A tanító adatok gyorsabb feldolgozását már egy adatbetöltő segíti. Méretezés, színtér-beállítások és mozaiknövelés szempontjából is továbbfejlesztett változat.

7. fejezet

Eredmények kiértékelése

7.1. Hibastatisztika

A tanítási és tesztelési folyamatoknál az egyes hibák száma, eloszlása lényegesen eltér egymástól, mely befolyásolhatja a modell pontosságát. Az egyes hibákhoz tartozó eloszlást a következő ábra szemlélteti:



7.1. ábra. Hibák eloszlása

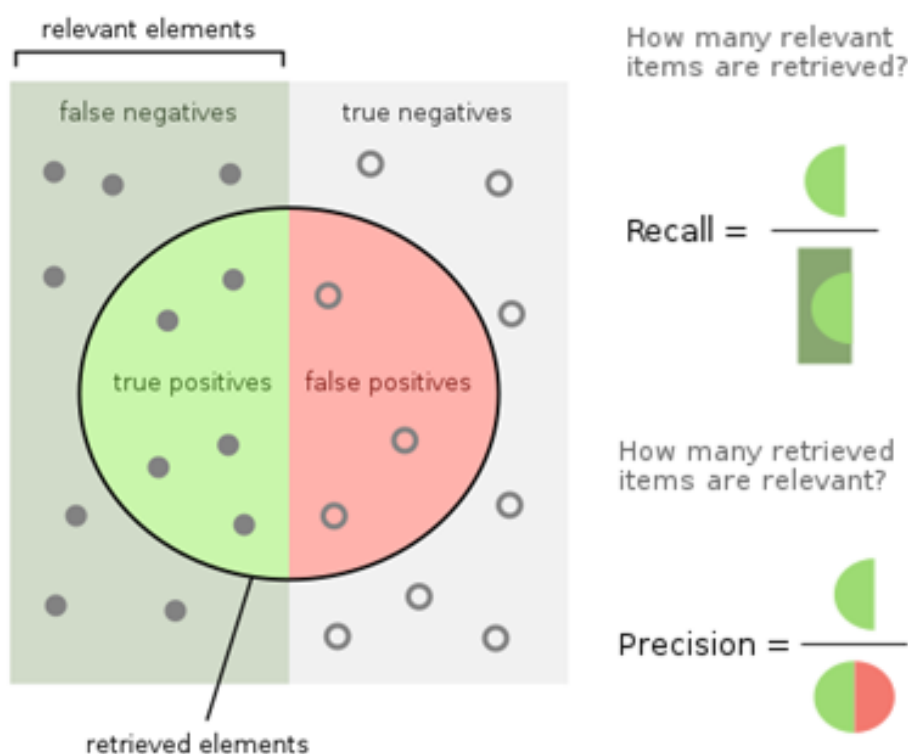
Jól látható, hogy bizonyos hibákból, pl. a *MicroCrack*/*Mikrorepedés*-ből lényegesen több van. Az ilyen hibákat a neurális hálózat a detekció során könnyebben megtalálja, viszont például a *Scratch*/*Karcolás* mennyiségéből adódó bizonytalanság könnyen szemügyre vehető a később bemutatandó eredmény táblázatok adatain. Ezt a problémát több megoldással is kezelni lehet: például augmentációval, vagy éppen változatosabb tanítóminták felvételével. A veszteségfüggvény és a modell architektúra optimalizálása is pontosabb ered-

ményt tud biztosítani. A *Contamination/Szennyeződés* mennyiségileg hiába eredményezhet egy pontosabb modellt, viszont a nehézséget az ilyen hibákat tartalmazó minták változatossága adja: van, ahol több cellán halad át, van, ahol csak egy cellát fed le. Ezen még az összevonásából származó *OtherError/Egyéb hiba* osztály sem ad megoldást a detekciónál.

7.2. Fogalmak, metrikák

Az eredmények kiértékelésénél az mAP - mean Average Precision [23] értékek vizsgálata mutatja a fejlődést ,melyet a modellek finomhangolása során értem el. Az AP minőségi mérőszámot biztosít az egyetlen osztályba soroláshoz az összes visszahívás (recall) során, mely a precizitási visszahívási görbe alatti területnek tekinthető. Ekkor az mAP jelenti az AP-k átlagát egy többosztályos klasszifikáció során.

Az alábbi ábra az egyes fogalmak megértését szemlélteti:



7.2. ábra. Modell kiértékelése során használt fontosabb metrikák

A modell pontosságának meghatározása során a tesztminták tényleges osztálycímkejének (annotált GT label-ek) összehasonlítása történik a modell által a tesztmintákra adott becsült válaszokkal. Ezen adatokat kell összegezni az úgynevezett keveredési mátrixban, mely két osztály esetén a következőképpen nézhet ki:

7.1. táblázat. Keveredési mátrix

	Pozitív (becsült)	Negatív (becsült)
Pozitív (tényleges)	a	b
Negatív (tényleges)	c	d

$$Pontosság = \frac{a + d}{a + b + c + d} \quad (7.1)$$

Fontosabb mérőszámok és képletek - amelyekkel az egyes tanított modellek tulajdonságait jellemezzük - a következők voltak, melyeket a modell validációjához implementáltam:

- Érzékenység (Recall, True Positive Rate): a pozitívak helyesen felismert aránya.

$$TPR = Sensitivity = \frac{TruePositive}{TruePositive + FalseNegative} \quad (7.2)$$

- Sajátosság (Specificity, True Negative Rate): a negatívak helyesen felismert aránya.

$$TNR = Specificity = \frac{TrueNegative}{TrueNegative + FalsePositive} \quad (7.3)$$

- False Positive Rate: a tévesen pozitívnak becsült egyedek aránya a negatív egyedekből.

$$FPR = \frac{FalsePositive}{FalsePositive + TrueNegative} = 1 - Specificity \quad (7.4)$$

- False Negative Rate: a tévesen negatívnak becsült egyedek aránya a pozitív egyedekből.

$$FNR = \frac{FalseNegative}{FalseNegative + TruePositive} \quad (7.5)$$

- Pontosság (Precision): a valós pozitívként felismertek mekkora része volt valóban pozitív.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \quad (7.6)$$

- F-mérték (F-measure): a precision és recall értékeknek a harmonikus középértéke.

$$F - measure = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (7.7)$$

7.3. Eredmények

7.3.1. Első annotálás (~ 1300 db kép) utáni eredmények

A vizsgált hálózatok közül a RetinaNet és a YOLO különböző verzióit tanítottam és teszteltem. Az eredmények kiértékelésekor az egyes mAP értékekben láthatjuk a változást. A hálózatok tanítását és tesztelését 10-, illetve a napelem gyártó (CoreComm SI Kft.) által kért 5-hibaosztályra is elvégeztem. Megjegyzés: az eddig nem említett, de a későbbi táblázatokban is szereplő *Examined* címke az egyes EL-képeken lévő napelem panelek teljes régióját jelöli, nem pedig a konkrét hibát!

7.2. táblázat. Neurális hálózatok 10 hibaosztályra történő tesztelési eredményei

mAP	RetinaNet v0.51	RetinaNet v1.0	YOLOv3	YOLOv5
All	0.5602	0.5448	0.637	0.724
Examined	0.9907	0.9858	0.988	0.982
ShortCircuitString	1.0000	0.8333	0.995	0.995
ShortCircuitCell	0.9501	0.9524	0.98	0.946
Crack	0.7276	0.7583	0.714	0.734
MicroCrack	0.5868	0.6028	0.654	0.679
DarkSpot	0.2589	0.2628	0.417	0.417
UnknownError	0.0000	0.0000	0.00516	0.995
Scratch	0.3056	0.3231	0.562	0.457
LowPowerCell	0.6463	0.6137	0.645	0.589
Contamination	0.1358	0.1154	0.413	0.443

7.3. táblázat. Neurális hálózatok 5 hibaosztályra történő tesztelési eredményei

mAP	RetinaNet v0.51	RetinaNet v1.0	YOLOv3	YOLOv5
All	0.6177	0.6526	0.776	0.807
Examined	0.9943	0.9954	0.999	0.982
ShortCircuitString	0.8333	0.6667	1	0.995
ShortCircuitCell	0.6889	0.8067	0.725	0.856
Crack	0.5991	0.6951	0.71	0.784
MicroCrack	0.4503	0.5759	0.715	0.697
OtherError	0.1403	0.1761	0.505	0.526

Az előző két táblázatban látható eredményeket a következő konfigurációkkal sikerült elérnem: a képméret 640×640 pixel, a RetinaNet-nél 100 epoch, míg a YOLO esetében 300 epoch lépés alatt történt a tanítás. A YOLO esetén a tanítás sebességét is figyelembe véve a *yolov5l* (large) modellt választottam, amely 1 perces tanítást jelentett epoch-onként. A RetinaNet-nél a Resnet50 backbone modellt használtam. Minden tesztelésnél különböző IoU (Intersection-over-Union átfedés) értékeket is vizsgáltam: a $\text{IoU} = 0.5$ küszöb feletti eredmények láthatók a táblázatokban.

Kezdetben a tanítások során a RetinaNet 0.5.1 volt alkalmazva, amely azonban még alacsony pontosságot eredményezett ($\sim 56\text{-}61\%$), az 10-, és a 5-osztály esetében is. Megfigyelhető, hogy minél újabb verziót és eltérő technológiát alkalmaztam a tanítási és tesztelési folyamatok során, annál jobb eredményeket kapunk. Viszont az eredmények jól tükrözik a hibastatisztikában már bemutatott adatbázis kiegyenlítetlenségét: amiből a legtöbb tanító példány volt (pl.: a mikrorepedés), azt könnyebben fel tudjuk ismerni. A nehézséget a kis tanító példányszámú karcolás, folt, stb. okozza az 5-osztályos összevonásnál is, mert ott több hibatípus tartozik egyetlen osztályba, így a felismerés is nehézkessé válik a detektor számára. További javulást egyrészt az augmentáció segítségével várhatunk, mely során a képek előfeldolgozása és a további mintaképek generálása és bevonása történik. Másrészt egy neurális hálózati modell leszűkítésével és annak finomhangolásával (hiper-paraméterek beállításával) pontosabb eredményeket érhetünk el.

7.3.2. ELDDS-I: részleges annotálás (~ 620 db kép) utáni eredmények

Mivel a YOLOv5 hálózat konfigurációja biztosította a legjobb eredményeket, ezért ennek a modellnek a használatával és finomhangolásával foglalkoztam a továbbiakban. Az eredmények javulását az is segítette, hogy az adatbázison egy újbóli, többkörök ellenőrző annotálási folyamatot hajtottam végre.

Első körben 600 tanító és 20 tesztelő kép (ELDDS-I) lett újraannotálva a kisebb adatmennyiség feldolgozásának gyorsasága, és az általam adott eredmények jobb értelmezhetősége miatt. Ezeken a tanító-, és tesztelő képeken már a Kontraszt Korlátozott Adaptív Hisztogram-kiegyenlítést, vagy röviden

7.4. táblázat. YOLOv5 modellek tesztelése

mAP	All	Examined	ShortCircuitString	ShortCircuitCell	Crack	MicroCrack	Othererror			
							DarkSpot	Scratch	LowPowerCell	Contamination
yolov5l -img 640	0.77	0.995	0.995	0.995	0.875	0.817	0.585	0.333	0.778	0.558
yolov5l6 -img 640	0.734	0.993	0.497	0.995	0.855	0.786	0.631	0.447	0.995	0.41
yolov5x6 -img 640	0.68	0.995	0.172	0.995	0.884	0.788	0.571	0.361	0.878	0.477
yolov5x -img 640	0.772	0.995	0.995	0.995	0.916	0.818	0.622	0.246	0.854	0.503
yolov5l -img640	0.859	0.995	0.995	0.928	0.912	0.827	0.496			
yolov5x -img 640	0.842	0.995	0.995	0.926	0.846	0.817	0.47			

CLAHE algoritmust is implementáltam.

7.3.3. YOLOv5 modellek vizsgálata

A YOLOv5 verzió kiválasztása után próbáltam megtalálni, behangolni azt az optimális modell konfigurációt, amely a legjobb detekciós eredményt képes biztosítani a 640 pixeles képméret és az új annotációk esetén. A modellek komplexitásában, méretben, sebességben, pontosságban különböznek egymástól. A tanítások során a legszembetűnőbb az volt, hogy a YOLOv5 modellek "6-os" változatai nem értek el jobb eredmény elődjeikhez képest. A legjobb eredményeket a *yolov5l* és *yolov5x* modellek adták, a fenti 7.4 táblázat mAP értékeiből ez jól látható. A megfelelő modell kiválasztásánál a sebesség is fontos szempont volt: azt tapasztaltam, hogy a *yolov5x* modell tanítása kétszer olyan lassú, mint a *yolov5l*. Mivel a méréseim közel ugyanazt az mAP értéket adták, ezért optimálisabbnak tűnt a gyorsabb taníthatóságú modellt használnom, tehát a *yolov5l*-t.

A főbb problémát az OtherError, azaz az összevont kategóriába tartozó hibák mAP eredményei jelentik, melyen a pontosabb annotáció, több és jobb minőségű tanító kép, és a hibák augmentálása tud segíteni.

7.3.4. Modell paramétereinek vizsgálata és beállítása

A modell verziók és konfigurációk összehasonlítását a hiperparaméterek (argumentumok) analízise követte, amely során a lehetséges beállításokat vizsgáltam meg: az SGD-, illetve Adam-optimalizálókat.

Az SGD [24] (Stochastic Gradient Descent) optimalizáló az 1950-es években megjelent „Gradiens süllyedés” technológiát alkalmazza. A módszer úgy változtatja a modell paramétereit az egyes iterációk során, hogy az változásokat okozzon a célfüggvényben, és kiválasztja azt a módosítási értéket, amely csökkenti a hiba arányát. Ez a folyamat addig ismétlődik, amíg a költségfüggvény a minimumhoz nem konvergál. A sztochasztikus változata ennek a módszernek a teljes tanító/képi adatbázis felhasználása helyett az adatok véletlenszerű kiválogatásával egy kis méretű részhalmazt vizsgál meg. Ez az optimalizálás nemcsak a modell teljesítményében, hanem a tanítási folyamat során is megfigyelhető: az egyes tanítási lépésekben, ha csak kis mértékű (elhanyagolható) változás történik a pontosságban, akkor a tanítási folyamat leáll (divergencia).

Azonban az összetett tanítási problémáknál ezen gradiens módszerek hatékonysága csökkenhet. Ennek a problémának a megoldásaként jött létre az Adam-optimalizáló eljárás, mely az SGD előnyös tulajdonságait is felhasználja: a „Root Mean Square Propagation” és az „Adaptive Gradient” algoritmusokat. Számos publikáció azt állítja [25] viszont, hogy nem lehet általánosítani, mivel bizonyos körülmények között nem sikerülhet az optimális megoldást elérni ezzel az Adam-optimalizálóval. Ezért a megfelelő, az adott modell pontosságában és teljesítményében előrelépést mutató optimalizáló megválasztása az adott problémára specifikusan történik.

7.5. táblázat. SGD és Adam-optimalizálók összehasonlítása

							Othererror			
	All	Examined	ShortCircuitString	ShortCircuitCell	Crack	MicroCrack	DarkSpot	Scratch	LowPowerCell	Contamination
mAP										
10 classes yolov5l -img 640 -sgd	0.77	0.995	0.995	0.995	0.875	0.817	0.585	0.333	0.778	0.558
5 classes yolov5l -img 640 -sgd	0.859	0.995	0.995	0.928	0.912	0.827	0.496			
10 classes yolov5l -img 640 -adam	0.838	0.995	0.995	0.995	0.886	0.843	0.749	0.58	0.982	0.517
5 classes yolov5l -img 640 -adam	0.882	0.995	0.995	0.971	0.93	0.85	0.55			
5 classes yolov5l -img 1280 -adam	0.872	0.995	0.995	0.905	0.966	0.91	0.464			

A fenti táblázatban a legjobb eredményt adó *yolov5l* modellt és az Adam-optimalizálót használva megvizsgáltam, hogy milyen pontosság érhető el. Megvizsgáltam a nagyobb, 1280-as képmérettel történő tanítást is (ekkor a modell az eredeti bemenő képeket nagyobb méretre skálázza át, és azokkal tanul), de csak kismértékű (4%) javulást tapasztaltam. Amiatt azonban, hogy a nagyobb tanító képméret mellett közel 4-szeres tanítási idővel (5 perc/epoch) kell számolni, nem érdemes a nagyobb képmérettel dolgozni.

7.3.5. Veszteségfüggvény módosítása

A tanítási folyamat egyes lépései során a neurális hálózat a bemeneti réteg felől a kimeneti réteg irányába előre haladva számítja ki az adott hibát, amit veszteségfüggvénynek (loss function) nevezünk. Ez a függvény nagyban befolyásolja a modell pontosságát, mert minősíti a hálózatot, azt, hogy mennyit vesztenénk a valós értékekből, ha a jelenlegi súlyokat használnánk. Emiatt a fő cél ennek a függvénynek, azaz a veszteségének a minimalizálása. Az egyes hálózati modellekben különböző veszteségfüggvények léteznek: a YOLOv5 alap esetben a `BCEWithLogitsLoss()` Pytorch alapú kereszt entrópiát és a szigmoid függvényt használja.

A `BCEWithLogitsLoss` veszteségfüggvény egyetlen osztályban egyesíti a szigmoid réteget és a `BCELoss`-t. Ez a verzió numerikusan stabilabb, mint egy sima Sigmoid réteg, mivel a műveletek egy rétegbe történő kombinálásával kihasználjuk a log-sum-exp módszert a numerikus stabilitás érdekében.

Ha a redukció 'none'-ra van állítva, akkor a következő képlettel lehet a veszteséget meghatározni:

$$\ell(x, y) = L = \{l_1, \dots, l_N\}^\top, \quad (7.8)$$

$$l_n = -w_n [y_n \cdot \log \sigma(x_n) + (1 - y_n) \cdot \log (1 - \sigma(x_n))] \quad (7.9)$$

ahol N a batch-szám. Ha redukció is fel lett használva, akkor a következő képlet érvényes:

$$\ell(x, y) = \begin{cases} \text{mean}(L), & \text{if reduction} = \text{'mean'}; \\ \text{sum}(L), & \text{if reduction} = \text{'sum'} \end{cases} \quad (7.10)$$

Az optimális eredmény miatt azonban egy saját veszteségfüggvényt implementáltam, amit *FocalComboTversky*-nek neveztem el, mert ez a Focal Tversky és a Combo veszteségfüggvények keverékéből alakítottam ki. A Focal Tversky veszteségfüggvény [26] a Tversky egyik változata, amelyben a Focal Loss-ban megjelent ún. gamma-módosító tényező található. A Tversky függvény a ki-egyensúlyozatlan orvosi adatkészletek szegmentálására tervezték, ahol olyan konstansokat használtak fel, amelyek meg tudják azt mondani, hogy bizonyos hibák milyen mértékben változtatják meg a veszteségfüggvényt. Ezt az alfa és béta paraméterek teszik lehetővé, amelyek a becsült pozitívokat, illetve a becsült negatívokat nagyobb mértékben büntetik a veszteségfüggvényben, ahogy értékük nő. A Combo veszteségfüggvény [26] a következőkből épül fel: a Dice Loss és egy módosított keresztentropia függvény kombinációja, amely a Tversky veszteségfüggvényhez hasonlóan további paraméterekkel rendelkezik, amelyek jobban megbüntetik a becsült pozitívokat, vagy a becsült negatívokat.

Az implementált veszteségfüggvény segítségével jelentős javulást tudtam elérni a pontosságban (mAP), melyet a következő táblázatban szemléltetek:

7.6. táblázat. Veszteségfüggvények vizsgálata

							Othererror			
mAP v5lite-g.pt100 epoch - 640	All	Examined	ShortCircuitString	ShortCircuitCell	Crack	MicroCrack	DarkSpot	Scratch	LowPowerCell	Contamination
BCEWithLogitsLoss (eredeti)	0.868	0.996	0.995	0.966	0.927	0.878	0.448			
Tversky Loss	0.876	0.996	0.995	0.977	0.889	0.848	0.555			
Combo Loss	0.855	0.996	0.995	0.961	0.854	0.841	0.484			
FocalCombo Tversky Loss (saját)	0.879	0.996	0.995	0.976	0.904	0.848	0.552			

Ahogy látható az eredeti veszteségfüggvény (BCEWithLogitsLoss) is elfogadható eredményt ad, de a FocalComboTverskyLoss kombinált verzió implementálásával további 2%-os javulást sikerült elérnem. A függvény szerkezetét a következő programkód mutatja be:

```
ALPHA = 0.5, BETA = 0.5, GAMMA = 1
class FocalComboTversky(nn.Module):
    def __init__(self, weight=None, size_average=True):
        super(FocalComboTversky, self).__init__()
    def forward(self, inputs, targets, smooth=1,
alpha=ALPHA, beta=BETA, gamma=GAMMA, eps=1e-9):
        inputs = inputs.view(-1)
        targets = targets.view(-1)
        intersection = (inputs * targets).sum()
        dice = (2. * intersection + smooth) / (inputs.sum() +
                                                targets.sum() + smooth)

        inp = torch.clamp(inputs, eps, 1.0 - eps)
        out = - (ALPHA * ((targets * torch.log(inputs))
+ ((1 - ALPHA) * (1.0 - targets) * torch.log(1.0 -
                                                inputs))))

        TP = (inp * out).sum()
        FP = ((1-out) * inp).sum()
        FN = (out * (1-inp)).sum()
        Tversky = (TP + smooth) / (TP + alpha*FP + beta*FN +
                                smooth)

        FocalTversky = (1 - Tversky)**gamma
        return FocalTversky
```

7.3.6. ELDDSI+II: teljes annotálás (~ 1300 db kép) utáni eredmények

Az ELDDSI+II adatbázisban már 1240 annotált és újraellenőrzött tanító képet használtam fel. A korábbi optimális paraméterek megtartása mellett (FocalComboTversky veszteségfüggvény, 640×640 -es képméret, Adam-optimalizáló, CLAHE kontraszt kiegyenlítés) tanított új modellt 20 képpel teszteltem, mely lényegesen jobb eredményt adott a korábbi esetekhez képest:

7.7. táblázat. ELDDSI+II: 1240-képet tartalmazó tanítás eredményei

							Othererror			
							DarkSpot	Scratch	LowPowerCell	Contamination
mAP v5lite-g.pt100 epoch - 640	All	Examined	ShortCircuitString	ShortCircuitCell	Crack	MicroCrack				
YOLOv5 new 5 yolov5l -img 640- <i>adam</i>	0.882	0.995	0.995	0.971	0.93	0.85	0.55			
yolov5l, 640, 300 epoch, Adam FocalCombo Tversky Loss,1240 tr.-20 te.	0.909	0.995	0.995	0.979	0.937	0.92	0.628			

A fenti táblázatban jól látható, hogy a vizsgált 5-hibaosztály esetében a YOLOv5l modellen további javulást sikerült elérnem összesített pontosságban (mAP), ami egyrészt az új veszteségfüggvénynek, másrészt a további tanítómintáknak is köszönhető. Jelenleg közel 91%-os pontossággal működik a modellem: így tehát az eredeti 61%-os kiinduláshoz képest 30%-os fejlődést sikerült elérnem.

Az is látható azonban, hogy bizonyos nehezen kezelhető, összevont osztállyal (OtherError) még mindig problémák vannak, amely abból adódik, hogy a több hibatípus összevonása bizonytalanná teszi a modellt.

A panel gyártó által definiált magas prioritású (súlyos, kritikus) hibákat azonban a modellem nagy pontossággal tudja detektálni és felismerni.

A tesztelési méréseket két eszközön végeztem: laptopon és a tanszék által biztosított HP szerveren, így a modell működésének teljesítményét CPU-n és GPU-n is egyaránt tudtam megfigyelni. A méréseket a következő specifikációkkal végeztem:

- CPU: Laptopos környezetben - Intel(R) Core(TM) i5-7200U, 8 GB RAM és SSD társításával az inferencia ideje ~ 1500 msec volt képenként.
- GPU: HP szerver környezetben - NVIDIA QP6000 24 GB-os videokártya és SSD-vel társítva az inferencia ideje 20 msec volt képenként. A tanítás is ezen a szerveren történt, a jelenlegi legjobb eredményt adó konfiguráció esetén ez ~ 8 óra volt.

Ezek a futási eredmények a következő YOLOv5 paraméterekkel lettek konfigurálva:

- 1240 tanító 20 tesztelő kép
- 5 hibaosztályhoz tartozó osztálycímkék
- YOLOv5l.pt 46.5 millió paramétert tartalmazó modell
- 640x640-es képméret (tanítási sebesség miatt - 1 epoch/perc)
- 32 batch szám
- 300 epoch
- Adam optimalizáló
- Általam készített FocalComboTversky veszteségfüggvény

8. fejezet

Összefoglalás

A feladatom során egyrészt az Elektro-Lumineszcens (EL) technológiával készült napelem panelek adatbázisát vizsgáltam: felismertem a korábbi annotációk ellentmondásaiból eredő hibákat, mely alapján kijavítottam, valamint bővítettem új kézi annotációkkal az adatbázist, végül pedig többszörösen is ellenőriztem azokat. Az adatbázisban (ELDDS I+II) lévő képeken Kontraszt Korlátozott Adaptív Hisztogram - CLAHE kiegyenlítést hajtottam végre annak érdekében, hogy a képek életlenségéből, intenzitás különbségéből adódó hibákat kiküszöböljem. Megismerkedtem a mély-tanuláson alapuló sokrétegű neurális hálózatok működésével, felépítésével (például a gerinchálózattal, az aktivációs-, és veszteségfüggvényekkel, valamint optimalizáló eljárásokkal). Feltérképeztem a visszaterjesztéses tanítási folyamatot, és megismerkedtem különböző egylépcsős detektorok hálózatokkal, melyeket behatóbban is tanulmányoztam. Végül kétfajta neurális hálózatra fókuszáltam: a YOLO, és a RetinaNet hálózatokra. Ezeknek a hálózatoknak az eltérő bemeneti adatformátumára alakítottam át az általam elvégzett adatbázis annotációk kimenétét (az implementált felhasználói felületem segítségével különböző konverziós script-ek hívhatók meg illetve az annotációk jeleníthetők meg grafikusán). A vizsgálatok során végül a YOLOv5-ös verzióját és yolov5l modelljét használtam különböző optimalizálókat tesztelve (SGD, Adam). Létrehoztam egy kombinált veszteségfüggvényt, valamint a különböző képi-augmentálási kísérleteket is végeztem, illetve azok hatását teszteltem. A YOLOv5l modellt felhasználva különböző hiper-paraméterekkel konfiguráltam, hangoltam és teszteltem le a mély-tanulás hatékonyságát, az optimális teljesítményt (mAP pontosságot, tanítási sebességet) keresve. Ez a pontosság jelenleg 91%, mely az eredeti kiindulási feltételekhez képest mintegy 30%-os javulást jelent.

Továbbfejlesztési lehetőségek: A kutató munka során szerzett tapasztalataim alapján a neurális hálózatok tanítási eredményeiből kiindulva a pontosságot csak úgy lehet tovább javítani, ha az "OtherError" összevont hibaosztály mAP értékét valamilyen módszerrel növelni tudom. Ezen módszerek lehetnek például a YOLO modellre épülő egyedi architektúra kialakítása (rétegek-szerkezetek összeállítása), az így kapott architektúrához kapcsolódó veszteségfüggvény optimalizálása. A bemeneti tanító képekkel is számos probléma volt: nagy méretűk ellenére eltérő kontrasztviszonyú és életlen minőségű képekkel dolgoztam. Egy saját EL-tesztelő állomás kiépítésével, jobb minőségű, élesebb EL képeket tudnánk készíteni. Mindezek alapján több, és változatosabb tanító és tesztelő minták hozzáadásával lehetséges lenne tovább javítani a detektor hiba-felismerési pontosságát.

A *Neurális hálózatok alkalmazása a napelem panelek minőségellenőrzésére* című témát a jövőben is folytatni kívánom.

Irodalomjegyzék

- [1] Portfolio.hu, „A nap mellett a gázalapú áramtermelés is nőtt magyarországon, de a teljes termelés kétharmada karbonsemleges.” <https://www.portfolio.hu/uzlet/20220217/a-nap-mellett-a-gazalapu-aramtermeles-is-nott-magyarorszagon-de-a-teljes-termelés-ketharmada-karbonsemleges-527389>, Feb 2022. Utoljára felkeresve: 2022. Március.
- [2] J. Sandhu, „A concise history of neural networks.” <https://towardsdatascience.com/a-concise-history-of-neural-networks-2070655d3fec>, Apr 2019. Utoljára felkeresve: 2021. December.
- [3] Y.-c. Wu and J.-w. Feng, „Development and application of artificial neural network,” *Wireless Personal Communications*, vol. 102, no. 2, pp. 1645–1656, 2018.
- [4] SunPower, „What is solar energy and how do solar panels work?.” <https://us.sunpower.com/solar-array-definition>, Mar 2021. Utoljára felkeresve: 2021. November.
- [5] „How do solar panels work? the science of solar explained.” <https://solect.com/the-science-of-solar-how-solar-panels-work/>, Mar 2018. Utoljára felkeresve: 2021. November.
- [6] U. Otamendi, I. Martinez, M. Quartulli, I. G. Olaizola, E. Viles, and W. Cambarau, „Segmentation of cell-level anomalies in electroluminescence images of photovoltaic modules,” *Solar Energy*, vol. 220, pp. 914–926, 2021.
- [7] Zae-Bayern, „Zae-bayern/elpv-dataset: A dataset of functional and defective solar cells extracted from el images of solar modules.” <https://github.com/Zae-Bayern/elpv-dataset>

- [//github.com/zae-bayern/elpv-dataset](https://github.com/zae-bayern/elpv-dataset). Utoljára felkeresve: 2022. Január.
- [8] „Research centre: Technological development.” <https://www.tecnalia.com/en/home>, Mar 1970. Utoljára felkeresve: 2022. Január.
- [9] W. K. Pratt, *Digital Image Processing*. John Wiley & Sons, 2016.
- [10] „Histograms - 2: Histogram equalization.” https://docs.opencv.org/3.4/d5/daf/tutorial_py_histogram_equalization.html. Utoljára felkeresve: 2022. Március.
- [11] YuAo, „Yuao/accelerated-clahe: Accelerated contrast limited adaptive histogram equalization.” <https://github.com/YuAo/Accelerated-CLAHE>. Utoljára felkeresve: 2022. Március.
- [12] N. Hassan, S. Ullah, N. Bhatti, H. Mahmood, and M. Zia, „The retinex based improved underwater image enhancement,” *Multimedia Tools and Applications*, vol. 80, no. 2, pp. 1839–1857, 2021.
- [13] „Napcellásrendszerek termográfiai állapotfelmérése ii.” <https://pim-kft.hu/szakmai-segitseg/publikaciok/vl-2019-4-napcellas-rendszerek-termografiai-allapotfelmerese-2/>. Utoljára felkeresve: 2021. November.
- [14] B. AG, „Basler ace aca2040-25gmnr - area scan camera.” <https://www.baslerweb.com/en/products/cameras/area-scan-cameras/ace/aca2040-25gmnr/>, Mar 2022. Utoljára felkeresve: 2022. Március.
- [15] Darkpgmr, „Darkpgmr/darklabel: Video/image labeling and annotation tool.” <https://github.com/darkpgmr/DarkLabel>. Utoljára felkeresve: 2021. Szeptember.
- [16] „Vgg image annotator (via).” <https://www.robots.ox.ac.uk/~vgg/software/via/>. Utoljára felkeresve: 2021. Január.
- [17] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, „Focal loss for dense object detection,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

- [18] Fizyr, „Fizyr/keras-retinanet: Keras implementation of retinanet object detection..” <https://github.com/fizyr/keras-retinanet>. Utoljára felkeresve: 2021. Október.
- [19] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, „You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [20] Ultralytics, „Yolov5.” <https://github.com/ultralytics/yolov5>, 2021. Utoljára felkeresve: 2021. Október.
- [21] „Neural circuit.” https://en.wikipedia.org/wiki/Neural_circuit, Dec 2021. Utoljára felkeresve: 2021. November.
- [22] M. Carranza-García, J. Torres-Mateo, P. Lara-Benítez, and J. García-Gutiérrez, „On the performance of one-stage and two-stage object detectors in autonomous vehicles using camera data,” *Remote Sensing*, vol. 13, no. 1, 2021.
- [23] R. J. Tan, „Breaking down mean average precision (map).” <https://towardsdatascience.com/breaking-down-mean-average-precision-map-a462f623a52>, Mar 2019. Utoljára felkeresve: 2022. Március.
- [24] Synced, „Iclr 2019: Fast as adam and good as sgd - new optimizer has both.” <https://medium.com/syncedreview/iclr-2019-fast-as-adam-good-as-sgd-new-optimizer-has-both-78e37e8f9a34>, Mar 2019. Utoljára felkeresve: 2022. Április.
- [25] A. Karpathy, „A peek at trends in machine learning.” <https://karpathy.medium.com/a-peek-at-trends-in-machine-learning-ab8a1085a106>, Apr 2017. Utoljára felkeresve: 2022. Április.
- [26] Bigironsphere, „Loss function library - keras & pytorch.” <https://www.kaggle.com/code/bigironsphere/loss-function-library-keras-pytorch/notebook#Combo-Loss>, Jul 2021. Utoljára felkeresve: 2022. Április.