HOSSZÚ TÁVÚ TANULÁS MÉLY NEURÁLIS HÁLÓZATOKBAN

Bsc Szakdolgozat

Írta:

Vass Boros Lea

Matematika BSc

Alkalmazott Matematikus Specializáció

Témavezető:

Benkő Beatrix

Matematikai Intézet

Számítástudomány Tanszék

Eötvös Loránd Tudományegyetem

Természettudományi Kar

Budapest, 2024

Tartalomjegyzék

Köszönetnyilvánítás

Szeretném köszönetemet kifejezni témavezetőmnek, Benkő Beatrixnak, aki magas szakmai tudásával végig türelemmel és odafigyeléssel támogatta ennek a szakdolgozatnak a színvonalas elkészülését.

1. Bevezetés

Az emberi agy működésének lényeges alkotóeleme az információfeldolgozás. Ezen folyamat során az agy képes permanensen új információkat megismerni és eltárolni anélkül, hogy elfelejtené a korábbi ismereteket. A tanulás során az emberi agy felülvizsgál és kiegészít minden beérkező információt, mielőtt megtartaná azokat, ezzel rendkívül eredményessé és hatékonnyá téve az ismeretszerzést.

A mesterséges neurális hálókat az emberi agy bonyolult működésének modellezése céljából és annak mintájára alkották. Azonban a biológiai agy működésével ellentétben a jelenlegi gépi tanulás alapú módszerek nem feltétlenül érnek el figyelemre méltó teljesítményt újonnan érkező adatok folyamatos adaptálása során. Erre márpedig nagy szükség lenne, tekintve, hogy az ilyen számítási rendszerek napról napra növekvő információáramlásnak vannak kitéve. Ahhoz, hogy ezt a hatalmas mennyiségű adatot a gépi tanulás és mesterséges intelligencia alapú modellek hatékonyan tudják feldolgozni, az szükséges, hogy ezen módszerek képesek legyenek a tudás fokozatos elsajátítására, finomhangolására és felhasználására hosszú távon. Ez jelenleg megoldatlan, az ilyen modelleket legtöbbször egyszeri sok adaton való tanítás után már nehéz teljesen új feladatokra továbbtanítani teljesítményromlás nélkül. *Ezen képességet, hogy egy adott modell képes legyen folyamatosan újabb adatokon tanulni és új információt eltárolni a korábban szerzett ismeretek megőrzésével együtt, hosszú távú tanulásnak nevezzük.*

Ennek megvalósításában a probléma az, hogy még a legfejlettebb mély neurális hálózatokról is ismert, miszerint ezek a modellek, melyek az osztályozási feladatok során jeleskednek, küzdenek a szekvenciális feladat-tanulással, hiszen a változó adateloszlásokhoz való alkalmazkodáshoz a teljes adathalmazon újra kell tanulniuk. A szekvenciálisan érkező feladatok során viszont az adatpéldányok típusának vagy eloszlásának fokozatos frissítése vagy változása katasztrofikus interferenciához vezethet. *Ez katasztrofikus felejtést eredményez, mely azt jelenti, hogy a modell az új adatok megtanulása során a modell-frissítés miatt hibás kimenetet ad olyan adatokon, amiken korábban képes volt az elvárt kimenetet produkálni.*

A hosszú távú tanulás során két kritikus problémát különböztetünk meg, amelyek az előbb említett katasztrofikus felejtéshez vezethetnek. Az első eset, amikor a modell szekvenciális sorrendben tanul egy több osztályt tartalmazó adathalmazból. Ekkor az új osztály tanulásakor a modell gyakran elfelejti a már korábban megtanult osztályokhoz kapcsolódó ismereteket. A második eset során a modell a korábbihoz hasonló szekvenciális sorrendben tanul, viszont ekkor az adatok azonos osztályból származnak. Ekkor az új adatok okozta adateloszlás-változás miatt a modell nem képes megfelelően alkalmazkodni, ami ismételten a korábbi ismeretek felejtéséhez

és teljesítménycsökkenéshez vezethet. Ezek alapján, csak abban az esetben lehet elfogadható pontosságú teljesítményt elérni, ha a tesztelési adatok és a tanítási adatok azonos vagy nagyon hasonló eloszlásból származnak. Ez az elv a statisztikai tanulás elméletének egyik alapvető feltételezésén, a független és azonos eloszlású adatminták meglétén múlik. Ezen feltétel megsértése szintén katasztrofikus felejtéshez vezethet.

Mindezek alapján, a hosszú távú tanulás módszerének legfontosabb megoldandó problémája a stabilitás-plaszticitás dilemma, amely egy jelentős tulajdonságot követel a neurális hálózatoktól a katasztrofikus felejtés minimalizálása érdekében. Az az elvárás, hogy egy modell képes legyen az újonnan megszerzett tudás eltárolására anélkül, hogy elfelejtené a már korábban megtanult adatokat. Még pontosabban, a dilemmának két fő feladatát különböztetjük meg a nevéből adódóan. Az első annak a meghatározása, hogy egy rendszer milyen mértékben legyen adaptív az új információk integrálásához, míg a második feladat annak az eldöntése, hogy a rendszernek mennyire stabilnak kell lennie ahhoz, hogy elkerülje a konszolidált tudásba való erőteljes beavatkozást.

A szakdolgozatban ismertetésre kerülő hosszú távú tanulás ezen problémákra keresi a megoldást: általános célja olyan tanulási rendszerek létrehozása, amelyek képesek különböző osztályokból és eloszlásokból származó adatokon való szekvenciális tanuláskor új ismereteket szerezni, miközben megakadályozzák, hogy az újonnan kapott adatok befolyásolják a már korábban megszerzett tudást. A dolgozatban bemutatom a hosszú távú tanulás alapvető elvárásait és általános célkitűzéseit, a szekvenciális feladat-tanulás formális definícióját és a különböző tanulási szcenáriókat. Ismertetésre kerül több kiértékelési szempont, amelyek alapján ezek a módszerek összehasonlíthatóak. A második fejezetében a dolgozatomnak összefoglalom a szakirodalomban ismert hosszú távú tanulási módszereket, és ezek közül kettőt, az elasztikus súlykonszolidációt és a visszajátszás módszerét részletesen ismertetem. A leírt módszerek hatékonyságát képosztályozási feladatok tanulásán keresztül értékelem ki a dolgozat utolsó fejezetében. A függelékben részletesen bevezetésre kerülnek a mesterséges neurális hálók alapfogalmai, amelyek elengedhetetlenek a szakdolgozatban leírtak megértéséhez.

2. A hosszú távú tanulás alapjai

Ahogy már a bevezetésnél is említésre került, az emberi tanulás folyamatosan fejlődött a hosszú évek során, mindig alkalmazkodva az aktuális, dinamikusan változó tanulási környezethez. Azonban a modern gépi tanulási módszerek nem képesek az ehhez hasonló szekvenciális tanulásra, hiszen a változó adatkészleteken futtatott tanulás nagymértékű teljesítményromlást eredményezhet.

A hosszú távú tanulás célja tehát olyan módszerek fejlesztése, amelyek javítják a tanulási hatékonyságot és lehetővé teszik a tudástranszfert különböző feladatok között mély neurális hálózatokban.

2.1. Célok és elvárások

A hosszú távú tanulás azon elvárásra épül, hogy a modellek folyamatosan fejlődjenek növekvő mennyiségű adatbemenet esetén, a lehető legkisebb mértékű katasztrofikus felejtéssel.

Magát a problémát jellemzően a megoldástól elvárt tulajdonságok határozzák meg. Ellentétben a gépi tanulás általános működésével, a hosszú távú tanulás olyan változó bemenetekre összpontosít, melyeket gyakran külön feladatok sorozatára bont az egymás utáni megtanulásuk érdekében. A hosszú távú tanulás nehézsége különböző kritériumok alapján változhat, például a feladatok hossza, típusa vagy ismétlődése szerint.

Egy modellnek két kulcsfontosságú tulajdonsággal kell rendelkeznie [Hadsell et al., 2020] szerint: az előrehaladó tudástranszferrel, amely lehetővé teszi, hogy minden új feladaton javuljon a teljesítménye az előzőekhez képest, valamint a visszafelé történő tudástranszferrel, amely során egy korábbi feladat újralátogatásakor a modellnek jobb teljesítményt kell nyújtania, kihasználva az aktuális feladatból származó adatokat. Mindezen kritériumokat a modellnek úgy kell teljesítenie, hogy korlátozott hozzáféréssel rendelkezik a korábbi feladatokhoz, illetve korlátozott kapacitással rendelkezik az adatok tárolására és a modell méretének növelésére.

A [Van de Ven and Tolias, 2019] cikk alapján a hosszú távú tanulás három különböző alterületre bontható. Megkülönböztetünk feladat-, tartomány-, illetve osztályonként történő hosszú távú tanulást. Ezek a fejezetben bővebb ismertetésre kerülnek.

2.2. Elengedhetetlen kritériumok

A hosszú távú tanulás első és legfontosabb kritériuma a katasztrofikus felejtés és az ahhoz kapcsolódó beavatkozások mértékének minimalizálása. Azonban emellett

számos további kritikus fontossággal rendelkező szempontot figyelembe kell venni az alkalmazási területen.

A katasztrofikus felejtés minimálisra csökkentéséhez hozzátartozik azon jelenség fontossága, miszerint az új feladatokon történő tanulás nem csökkentheti a már korábban megtanult feladatok teljesítményét. Egy másik szempont, hogy a modellnek minimális hozzáféréssel kell rendelkeznie a korábban elvégzett feladatokhoz, melyhez hozzátartozik, hogy ezen korábbi feladatok tárolására csak korlátozott tárhely elérhető. Mindezek mellett a modell nem léphet interakcióba a korábbi feladatokkal.

A modellnek képesnek kell lennie gyorsan alkalmazkodni az új feladatokhoz és a bemenetváltásokhoz, miközben a korábbi feladatok ismételt bemutatásakor is gyors alkalmazkodásra van szükség. Továbbá törekedni kell a modell számítási kapacitásának és méretének minimális növekedésére, valamint a megközelítések skálázhatóságára is. Ez utóbbi jelentése, hogy nem lehet új modellt létrehozni minden új feladathoz egy régebbi feladat elvégzése után.

Egy másik különösen fontos, már a bevezetésben is említett probléma a plaszticitás megőrzése. Ennek kritériuma, hogy az új feladatok tanulásakor és megfigyelésekor a modell képes legyen ugyanolyan hatékonysággal tanulni, mint azelőtt. Emellett a tanulás hatékonysága és a teljesítmény növelése érdekében a modellnek egy feladat megtanulásakor tudnia kell javítani az adott feladathoz kapcsolódó már megtanult és jövőbeli feladatok teljesítményén. Utolsó sorban egy apró, azonban nem elhanyagolható kikötés, miszerint a tanulás során a modell nem támaszkodhat ismert feladatcímkékre vagy feladathatárokra.

2.3. A pontos definíció

Jelölje a folyamatos tanulás előtti kezdeti modellt $F^0_{\boldsymbol{\theta}}(\boldsymbol{x})$. Egy folyamatos tanulási osztályozási probléma $T \in \mathbb{N}$ feladat sorozataként definiálható és minden $t \in \{1, \ldots, T\}$ feladat során hozzáférünk egy D^t adathalmazhoz, amely független és azonos eloszlású mintákból áll: $D^t = \{(\boldsymbol{x}^t_i, y^t_i) \,|\, i = 1, 2, \ldots, |D^t|\}$, ahol $\boldsymbol{x}^t_i$ jelöli az i. megfigyelt adatponot, y^t_i pedig a hozzá tartozó címkéket. Ezen adathalmazok adatfolyamát jelölje D, ahol $D = D^1, D^2, \ldots, D^T$.

Tegyük fel, hogy minden t feladat hozzá van rendelve különböző osztályokhoz, melyet Y^t jelöl, és $Y^t = \{y^t_i \mid \forall y^t_j \neq y^t_i, 1 \leq j < i \leq |D^t|\}$. Fontos megjegyezni, hogy minden feladat osztályai különböznek. Ezek alapján $\sum_{t=1}^{T} |Y^t| = C$ az összes osztály száma.

Ha a modellt D^t megtanulása után $F^t_{\boldsymbol{\theta}}(\boldsymbol{x})$-vel jelöljük, akkor az általános hosszú távú tanulási feladat [Qu et al., 2021] szerint definiálható P-vel, ahol $P = P_1, P_2, \ldots, P_T$ és $P_t : \langle F^{t-1}_{\boldsymbol{\theta}}(\boldsymbol{x}), D^t \rangle \rightarrow F^t_{\boldsymbol{\theta}}(\boldsymbol{x}), \forall\, t \in 1, \ldots, T$.

Ezen jelölésekkel a hosszú távú tanulási módszer célja, hogy minden P_t folyamaton tanult $F_{\boldsymbol{\theta}}^t(\boldsymbol{x})$ modell jó teljesítményt érjen el az új feladaton az előző feladatokon elért teljesítmények rontása és befolyásolása nélkül. Fontos megjegyzés, hogy a külön memóriát használó módszerek esetén a D^t előtti adathalmazokból tárolt adatpéldányok elérhetőek a P_t folyamat során a memórián keresztül, egyébként nem.

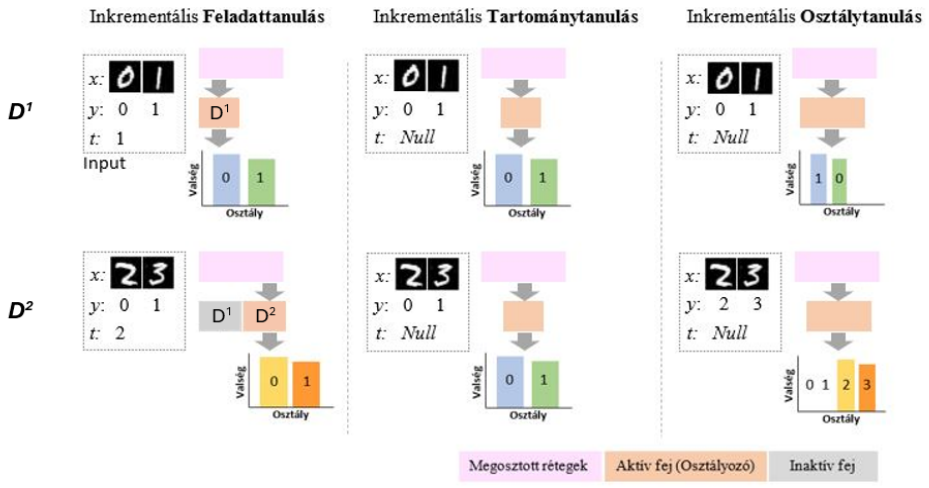
2.1. Példa. A definíció pontos megértése érdekében vegyük példaként a CIFAR10 halmaz [Krizhevsky et al., 2009] 5 feladatra bontását.

Esetünkben öt feladatra bontjuk a tanítást, így a feladatok adatfolyama, vagyis D a következőképpen néz ki: $D = \{D^1, D^2, D^3, D^4, D^5\}$. Nézzük most az első feladatot: ekkor $D^1 = \{(\boldsymbol{x}_1^1, y_1^1), (\boldsymbol{x}_2^1, y_2^1)\}$. Ezek a D^1 feladatban megtanulandó osztályokhoz tartozó tanító példák, tehát $\boldsymbol{x}_1^1$ és $\boldsymbol{x}_2^1$ az első és második osztályhoz tartozó képbemenetek, az y_1^1 és y_2^1 pedig a hozzájuk tartozó címke bemenetek. A D^1 feladat elvégzése után D^2 következik hasonló módon, majd a többi így tovább.

2.4. Különböző tanulási szcenáriók

Ahogy korábban már említésre került, a hosszú távú tanulás három különböző forgatókönyvre bontható: megkülönböztetünk feladat-, adattartomány-, illetve osztály-inkrementális hosszú távú tanulást [Hsu et al., 2018] szerint.

Az 1. ábra a Split MNIST első két feladatának tanulását prezentálja. A Split MNIST egy olyan feladatsorozat, amelyet gyakran használnak a folyamatos tanulási módszerek értékelésére. A hagyományos MNIST [Deng, 2012] adatbázis felhasználásával a Split MNIST öt különböző feladatra osztja fel az adatokat, ahol minden feladat 2 számjegyből áll [Van de Ven et al., 2022]. Az összes felosztás tehát $0/1, 2/3, 4/5, 6/7, 8/9$ lenne, itt csupán a D^1 és D^2 esetét nézzük.



1. ábra. A Split MNIST által generált három hosszú távú tanulási szcenárió

Az ábrán a tanítás bemeneteit a szaggatott téglalap jelöli, ahol x, y, t sorra a bemeneti képeket, a célosztály azonosítóját és a bemeneti feladat azonosítóját definiálja. A téglalap mellett a neurális hálózat modellje szerepel az előrejelzett $P(Y^t)$ kimenettel, a különböző "dobozok" színei pedig a modell osztályonkénti kimeneteinek felelnek meg.

Inkrementális feladattanulás

A feladat-inkrementális tanulás a legegyszerűbb szcenárió a három közül. Az inkrementális feladattanulás esetén a modell feladata az, hogy különböző, egymástól elkülönült feladatokat tanuljon meg egymás után. Ezen tanulás alapelve, hogy a modell megkapja a feladatazonosítót a tesztelés során. Az inkrementális feladattanítás feladatspecifikus módszerekkel történik, minden feladat saját kimeneti egységgel rendelkezik, miközben a hálózat többi része közös minden feladatra. Ezen hálózati architektúrát "többfejű" kimeneti rétegűnek is nevezik.

Ezt a tanulást két feladatra definiáljuk a már fent bevezetett alakban:

$$P_1 : \langle F^0_{\boldsymbol{\theta}}(\boldsymbol{x}), D^1 \rangle \rightarrow F^1_{\boldsymbol{\theta}}(\boldsymbol{x}) \text{ és } P_2 : \langle F^1_{\boldsymbol{\theta}}(\boldsymbol{x}), D^2 \rangle \rightarrow F^2_{\boldsymbol{\theta}}(\boldsymbol{x}),$$
$$\text{ahol } Y^1 \cap Y^2 = \varnothing \text{ és } Y = Y^1 \cup Y^2.$$

Ezt az esetet a 1. ábra első oszlopa szemlélteti. Ekkor az osztályok különbözőek, továbbá $P(Y^1) \neq P(Y^2)$ is teljesül. Ezen esetben a neurális hálózat minden feladathoz egyedi kimeneti réteggel rendelkezik, és tesztelés idején csak a feladatnak megfelelő kimeneti réteg lesz használva.

Inkrementális tartománytanulás

A tartománytanulás esetében nincs szükség feladatazonosító megadására a teszt idejében, ugyanis ezen tanulás célja az egyes feladatokon belüli osztályok megkülönböztetése a feladatok megkülönböztetése helyett. Ekkor a modellnek csak az adott feladatra kell összpontosítania, ezáltal nem szükséges kikövetkeztetnie pontosan melyik feladatról is van szó. Tehát ezen tanulás esetén a modell különböző adatforrásokból származó adatokat tanul meg, de a feladat és a kimeneti osztályok azonosak maradnak és a modell célja, hogy alkalmazkodjon a különböző adatforrásokhoz anélkül, hogy az előzőleg megtanult feladatok teljesítménye romlana. Ezt a következőképpen definiáljuk:

$$P_1 : \langle F^0_{\boldsymbol{\theta}}(\boldsymbol{x}), D^1 \rangle \rightarrow F^1_{\boldsymbol{\theta}}(\boldsymbol{x}) \text{ és } P_2 : \langle F^1_{\boldsymbol{\theta}}(\boldsymbol{x}), D^2 \rangle \rightarrow F^2_{\boldsymbol{\theta}}(\boldsymbol{x}),$$
$$\text{ahol } Y^1, Y^2 \subseteq Y \text{ és } Y = Y^1 \cup Y^2.$$

Ekkor a modell ugyanazt az Y^1 és Y^2 kimeneti teret biztosítja. A feladat azonosítója szükségtelenné válik, a bináris osztályozás kimeneti eloszlása azonos ($P(Y^1) = P(Y^2)$), és a $\{0, 1\}$ címke képei a 0/1 számjegyről 2/3-ra váltanak.

Inkrementális osztálytanulás

Az osztálytanulás célja nevéből adódóan az osztályok megkülönböztetése mind a feladatokon belül, mind a feladatok között. Az osztály-inkrementális jelző az új osztályok fokozatos tanulásának általános problémáját jelöli. A modellnek tehát képesnek kell lennie megoldani az eddig látott feladatokat, továbbá ki kell következtetnie, éppen melyik feladat adatait kell feldolgozza. Ezt a következőképpen definiáljuk:

$$P_1 : \langle F^0_{\boldsymbol{\theta}}(\boldsymbol{x}), D^1 \rangle \to F^1_{\boldsymbol{\theta}}(\boldsymbol{x}) \text{ és } P_2 : \langle F^1_{\boldsymbol{\theta}}(\boldsymbol{x}), D^2 \rangle \to F^2_{\boldsymbol{\theta}}(\boldsymbol{x}),$$
$$\text{ahol } Y^1 \cap Y^2 = \varnothing \text{ és } Y = Y^1 \cup Y^2.$$

Az utolsó forgatókönyben a feladatok közti kimeneti terek diszjunktak, vagyis $Y^1 \neq Y^2$, a 1. ábrán ezt az esetet a harmadik oszlop szemlélteti. Az osztályok különbözőek. Ezen példában a többosztályú tulajdonság miatt a $P(Y^1) \neq P(Y^2)$ következmény teljesül. Ez a tanulás-típus abban különbözik az inkrementális feladattanulástól, hogy itt a modellnek egyetlen feladatok között megosztott kimeneti része lehet csak, amely vagy előre rögzített számú kimenete adhat csak, vagy pedig dinamikusan bővíthető.

2.5. A független és azonos eloszlású feltételezés

Egy optimális modell a működéséhez hatalmas mennyiségű adatot vesz figyelembe egyszerre, miközben az adatokkal való párhuzamos számolásra is törekszik. Ennek kivitelezéséhez egy hálózat paramétereit úgy kell beállítani, hogy a veszteségfüggvény minimális legyen. Azonban ehhez egyenként kellene beállítani a paramétereket, ami egy nagyobb hálózatnál már rendkívül sok időbe telne.

Ezen probléma megoldására lett bevezetve a gradiens alapú módszer, mely a paraméterek külön-külön történő módosítása helyett a hálózat összes paraméterét egyszerre változtatja. Ez a módszer iteratív, és minden lépésben a paraméterek frissítésével csökkenti a veszteséget, amíg el nem éri az optimális értéket.

A gradiens alapú tanulás hatékonyságát és gyorsaságát a kötélhúzási dinamika biztosítja, ahol a paraméterek folyamatos kisebb-nagyobb értékekre való módosításával iteratívan közelítjük meg az optimális megoldást. A gradiensek átlagolásánál a frissítés után láthatóvá válik, hogy mely adatminták adnak jobb eredményt és melyek nem, így a tanulás hatékony és gyors fejlődést érhet el.

A gradiens alapú tanulás működését egy konkrét példával illusztrálhatjuk. Tegyük fel, hogy van egy kép. Ahelyett, hogy megvizsgálnánk egy paraméter gradiensét és annak hatását az adott paraméterre a hibafüggvény szerint, kiátlagoljuk a gradienseket az összes adatra és a teljes neuronhálóra. Ez azt jelenti, hogy a hibát visszaterjesztjük a hálózaton keresztül. Ez szükséges, mert ha a hálózat minden paraméterét külön-külön vizsgálnánk meg minden egyes képre, a tanulási folyamat rendkívül időigényes és gyakorlatilag kivitelezhetetlen lenne. Ezért hatékonyabb eljárás a teljes hálózatra egyszerre kiterjeszteni a gradienseket. Azonban itt felmerül a kérdés, hogy mi alapján történjen a módosítás? Egyes adatpontok növelnék, míg mások csökkentenék ugyanazt a paramétert, innen adódik a kötélhúzó dinamika fontossága.

A független és azonos eloszlású feltételezés (i.i.d.) döntő szerepet játszik a mély neurális hálózatok képzésében és általánosításában. Az i.i.d. azt jelenti, hogy minden adatminta független a többitől, és azonos eloszlás szerint oszlik el. Ha a betanítási adatok strukturált mintákat tartalmaznak, a modell túlságosan illeszkedhet ezekhez a specifikus mintákhoz, és nehezen tud jól általánosítani a még nem látott adatokra. Az i.i.d. segít az eloszlás rögzítésében, lehetővé téve, hogy a modell robusztusabb és általánosíthatóbb reprezentációkat tanuljon meg.

2.6. Kiértékelési metrikák

Ahogy az már korábban párszor említésre került, a hosszú távú tanulás hatékony működéséhez elengedhetetlen fontossággal bír az új feladatok gyors megtanulása, továbbá a katasztrofikus felejtés minimalizálása. Ezek elérése érdekében érdemes mérni a tanulás új feladatokon nyújtott teljesítményét és a korábbi tudás elfelejtésének mértékét. Ezen fejezet a legfontosabb ilyen kiértékelési metrikákat mutatja be. Definiálja $a_{p,t}$ a p-edik teszthalmaz pontosságát t feladat hosszú távú tanulása után. Feltesszük, hogy $p \leq t$.

2.2. Definíció. Átlagos pontosság. (Acc)

A hosszú távú tanulás teljesítményét méri t darab feladat megtanulása után, melyet a következő képlet ad meg:

$$Acc_t = \frac{1}{t} \sum_{p=1}^{t} a_p^t.$$

2.3. Definíció. Átlagos felejtés. (Fg)

Megadja, hogy mennyi tudás felejtődött el az első $t-1$ feladat megtanulása során. Ezt a következő képlet definiálja:

$$Fg_t = \frac{1}{t-1}\sum_{p=1}^{t-1} f_p^t\,,$$

ahol

$$f_p^t = \max_{o\in\{1,\ldots,t-1\}} a_{o,p} - a_{t,p}, \forall p < t.$$

Vagyis ahhoz, hogy az f_p^t-val jelölt p feladatnál felmerülő tudásvesztés mennyiségét megkapjuk, a hosszú távú tanulás során megszerzett maximális tudás és a t feladat megtanulása után megmaradó tudás közötti különbséget kell venni.

2.4. Definíció. Adaptálás-képtelenség. (I_t)

Azon jelenség mértékét definiálja, hogy az általános kötegelt tanuláshoz képest a hosszú távú tanulás mennyire akadályozza a modellt egy új feladat megtanulásában. Ezen mértéket a következő képlet adja meg:

$$I_t = a_t^\star - a_{t,t},$$

ahol $a_t^\star$ a t-edik feladathalmaz pontosságát jelöli abban az esetben, ha t feladat elvégzéséhez kötegelt tanulás van használva.

2.5. Definíció. Visszafele történő tudástranszfer. (BWT_t)

Azon mértéket adja meg,hogy a t-edik feladaton végzett hosszú távú tanulás milyen mértékben befolyásolja a korábban tanult feladatok elvégzését. Ezt a következőképpen definiáljuk:

$$BWT_t = \frac{1}{t-1}\sum_{p=1}^{t-1}(a_{p,t} - a_{p,p}).$$

2.6. Definíció. Jövőbeli tudástranszfer. (FWT_t)

Azon mértéket adja meg,hogy a t-adik feladaton végzett hosszú távú tanulás milyen mértékben befolyásolja potenciálisan a jövőbeli feladatok teljesítését. Az erre szolgáló képlet:

$$FWT_t = \frac{1}{t-1}\sum_{p=2}^{t}(a_{p-1,p} - b_p),$$

ahol b_p a p-edik feladat teszthalmazának pontossága véletlenszerű inicializáláskor.

2.7. Más gépi tanulási területekkel való kapcsolat

Ezen fejezet a gépi tanulás különböző területeit ismerteti. A tudásmegosz-tás és az adaptáció gondolatai már mindezen területeken sikeresen kialakultak, azonban az egyes területeken eltérő módokon. A következő alfejezetekben a különböző területeket röviden ismertetjük, majd kiemeljük a legfontosabb eltéréseket a hosszú távú tanulással [Aljundi, 2019] szerint.

Meta-tanulás

A meta-tanulás folyamatát gyakran a ”tanulás tanulása”-ként is emlegetik. Ez utóbbi meghatározás arra utal, hogy a tanulási magatartás hogyan javítható a tanító adatok révén. A meta tanulás célja, hogy egy modellt megtanítson arra, hogyan tanuljon új feladatokat hatékonyan, azaz hogyan adaptálódjon gyorsan új feladatokhoz minimális adattal vagy néhány tanulási lépéssel.

Ekkor tehát egy adott feladatsoron a modell a múltbeli feladatokból szerzett ismereteket használja fel a jelenlegi feladatok elvégzésére és javítására. Azonban elegendő mennyiségű metaadat használatára van szükség ahhoz, hogy a modell megértse, hogyan válhat az automatikus tanulás rugalmassá a problémák megoldásában, ezzel javítva a meglévő algoritmusok teljesítményét. Ezen metaadatok a korábban tanult feladatok ismereteit tartalmazzák és az új feladat hatékony kidolgozására szolgálnak. A meta tanulás során a modell a tanulási algoritmust tanulja meg, és nem pusztán a feladatok specifikus paramétereit.

A hosszú távú tanuláshoz hasonlóan a modell fő feladata, hogy a lehető leghatékonyabban alkalmazkodjon egy nagyszámú képzési feladathoz, miközben a képzési adatok véletlenszerűen vannak kinyerve egy osztályból és a tesztadatok csupán néhány példával vannak megadva. Azonban a hosszú távú tanulástól eltérő módon, a meta-tanulás minden kontaminációs adattal egyszerre rendelkezik, így nem tudja megakadályozni a korábban tanult tudás elfelejtését, vagyis felléphet a katsztrofikus felejtés problémája.

Online tanulás

A hagyományos offline tanulási technikákkal ellentétben – ahol a modell egyszerre tanulja meg a teljes adathalmazt –, az online tanulás során az adatok szekvenciális sorrendben válnak elérhetővé. Ebben az esetben az adatfolyamot $D = D^1, D^2, \ldots, D^T$ jelöléssel illetjük, ahol minden D^t egy adott időpontban elérhető adathalmaz. Az online tanulás célja a modellek szekvenciális optimalizálása $P = P_1, P_2, \ldots, P_T$ folyamatok segítségével, ahol minden $P_t : \langle F_{\boldsymbol{\theta}}^{t-1}(\boldsymbol{x}), D^t \rangle \rightarrow F_{\boldsymbol{\theta}}^{t}(\boldsymbol{x})$ lépésben a modell

frissül az újonnan érkező adatok alapján, hogy minden lépésben a lehető legjobb előrejelzőt biztosítsa a jövőbeli adatokhoz.

Ezek a törekvések a hosszú távú tanulásnak is fontos jellemzői. Azonban a hosszú távú tanulással ellentétben az online tanulás továbbra is a független és azonos eloszlású adatmintavételi folyamaton és egyetlen feladaton alapul.

Transzfer tanulás

A transzfertanulás az egyik feladat során megszerzett tudás tárolására összpontosít, miközben egy másik, de kapcsolódó feladatot próbál megoldani. Formálisan, ha adott egy forrásadat D^S adathalmazzal és a hozzá tartozó P_S feladattal, illetve egy céladat D^T adathalmazzal és P_T feladattal, akkor az átviteli tanulás célja a P_T feladat tanulásának támogatása D^T adathalmazzal, D^S és P_S használatával, ahol $D^S \neq D^T$ és $P_S \neq P_T$.

A hosszú távú tanuláshoz hasonlóan a transzfertanulás is a korábban megtanult adateloszlás hasznosítására fókuszál, azonban a hosszú távú tanulással ellentétben a transzfertanulás a célfeladat megtanulása után nem képes folyamatosan adaptálni a feladatokat.

Többfeladatos tanulás

A többfeladatos tanulás több tanulási feladatot is képes egyidejűleg megoldani megosztott paraméterek halmazának vagy részhalmazának felhasználásával, miközben kihasználja a feladatok közötti hasonlóságokat és különbségeket. Formálisan, a többfeladatos tanulás célja, hogy egyszerre több osztályozási feladat teljesítményét javítsa. Jelölje $F_{\boldsymbol{\theta}}^{MTL}(\boldsymbol{x})$ a többfeladatos tanulási modellt, amelyet a $D = D^1, D^2, \ldots, D^T$ adathalmazon tanítunk. A többfeladatos tanulási folyamat P^{MTL}-vel jelölhető, ahol $P^{MTL} : \langle F_{\boldsymbol{\theta}}^0(\boldsymbol{x}), D\rangle \rightarrow F_{\boldsymbol{\theta}}^{MTL}(\boldsymbol{x})$.

A modellek külön-külön történő betanításával ellentétben ez a tanulási módszer jobb hatékonyságot és pontosságot eredményez, mint a feladatspecifikus modellek. Ezen felül az algoritmus, mely megakadályozza a többfeladatos tanulás túlillesztését működik a többfeladatos tanulásnál, mert minden bonyolultságot egységesen büntet.

A fő különbség a többfeladatos tanulás és a hosszú távú tanulás között az, hogy az az előbbi követi az összes feladat offline betanítását az összes feladatadat egyidejű jelenlétében, illetve a többfeladatos modell bevezetése után nem jár semmilyen adaptációval, ellentétben a hosszú távú tanulással.

3. Hosszú távú tanulási módszerek

A hosszú távú tanulás egy tanuló rendszer – legyen az biológiai vagy mesterséges neurális hálózat – azon képességére utal, mely a szekvenciálisan történő ismeretszerzésre összpontosít a már korábban megszerzett tudás elfelejtése nélkül. A mesterséges neurális hálók ilyen jellegű tanítása azzal a problémával jár, miszerint új, más eloszlásból származó adatokon való tanításukkor a már korábban tanult feladatokon a teljesítményük drasztikusan romlik a paraméterváltoztatás miatt. Ennek megelőzésére számos módszert és technikát fejlesztettek ki, melyek három fő csoportba sorolhatóak [Parisi et al., 2019] szerint:

1. **Regularizációs technikák**

   A regularizáció alapú módszerek korlátozzák a különböző modellparaméterek frissítését, ezáltal stabilizálják a korábban tanult ismereteket és csökkentik a katasztrofikus felejtés mértékét. Ezen módszerek a katasztrofikus felejtést a tanítás során a veszteségfüggvényhez adott regularizációs hibafüggvénytaggal próbálják minimalizálni. Ezen regularizációs tag a korábbi feladatokon meghatározott optimális súlyok változását bünteti, ezáltal korlátozva a modell képességét, hogy jelentősen eltérjen a korábban optimalis állapottól. A veszteségfüggvény ezen megszorítása arra ösztönzi a modellt, hogy egyszerűbb és robusztusabb reprezentációkat tanuljon meg. Az egyik leghatékonyabb és leggyakrabban használt regularizációs technika az elasztikus súlykonszolidáció (Elastic Weight Consolidation), avagy EWC, mely szabályozza a súlyfrissítéseket a korábban megtanult feladatok szerinti fontosságuk alapján. Ezen módszer a 3.1 fejezetben hosszasan ismertetésre kerül.

2. **Visszajátszáson alapuló módszerek**

   A visszajátszáson alapuló módszerek a katasztrofikus felejtés problémáját múltbeli feladatok adatmintáinak tárolásával és újrafelhasználásával vagy a tanítás során mesterségesen generált minták tárolásával, használatával kezelik. Ezen módszerek a tanítás során külön memóriát használnak a korábbi adatok vagy éppen egy azokat generáló modell paramétereinek tárolására az új adatok tanulása mellett. Ennek a módszernek az egyik leggyakrabban használt változata, amely a memória feltöltéséhez az úgynevezett reservoir mintavételezést [Vitter, 1985] alkalmazza, a 3.2 fejezetben kerül részletesebb bevezetésre.

3. **Architekturális megközelítések**

   Ezen módszerek úgy próbálják meg kezelni a katasztrofikus felejtés problémáját, hogy a modell struktúráját dinamikusan változtatják és az új tanulási felada-

tokhoz igazodva bővítik, vagy pedig csak bizonyos hálórészeket használnak adott feladat tanulásához. A progresszív neurális hálózatok [Rusu et al., 2016] például új neuronokat adnak a neurális hálóhoz egy új feladat tanulásakor, amelyek kizárólag az aktuálisan tanult feladatra specializálódnak. Egy másik sikeres módszer, hogy minden feladathoz alkalmasan válasszuk ki a teljes modell súlyainak egy kis hányadát, amelyekkel jó teljesítmény érhető el az adott feladaton [Wortsman et al., 2020].

Ezen három fő kategória módszereinek a kombinálása is lehetséges a legjobb eredmények elérése érdekében, például lehet gradienseket tárolni a memóriában a konkrét tanítópéldák helyett [Farajtabar et al., 2020] vagy akár lehet visszajátszást és regularizációt egyszerre használni a tanítás során. A 4. fejezetben az utóbbira vonatkozó kísérleteket mutatok be, a visszajátszás és az elasztikus súlykonszolidáció módszerek kombinálásával.

3.1. Elasztikus súlykonszolidáció (EWC)

Ahogy már a bevezetésben is említésre került, az Elastic Weight Consolidation, avagy EWC egy olyan regularizációs technika, amely hatékonyan alkalmazható több feladat szekvenciális tanulására [Kirkpatrick et al., 2017], [Aich, 2021].

Az EWC alapvető koncepciója abban rejlik, hogy fontossági súlyokat rendel a paraméterekhez és a korábbi feladatok megoldása szempontjából fontos paraméterek változtatását korlátozza. Egy új feladat megtanulása során az EWC egy regularizációs kifejezést vezet be, melynek feladata, hogy szabályozza a fontos paraméterek megváltoztatását és segítse a korábbi feladatokból kinyert információ megőrzését.

A fontos paraméterek szelektív korlátozásával az EWC egyensúlyt biztosít a plaszticitás és a stabilitás között, ezzel lehetővé téve a hálózat számára az új feladatok hatékony tanulását és a múltbeli feladatokból származó tudás megőrzését egyaránt.

Tekintsünk két egymást követő feladatot, a hozzájuk tartozó megtanulandó adathalmazok: D^t, D^{t+1}. Ezeket ebben a fejezetben az átláthatóbb definíciók és képletek érdekében jelöljük úgy, hogy A és B.

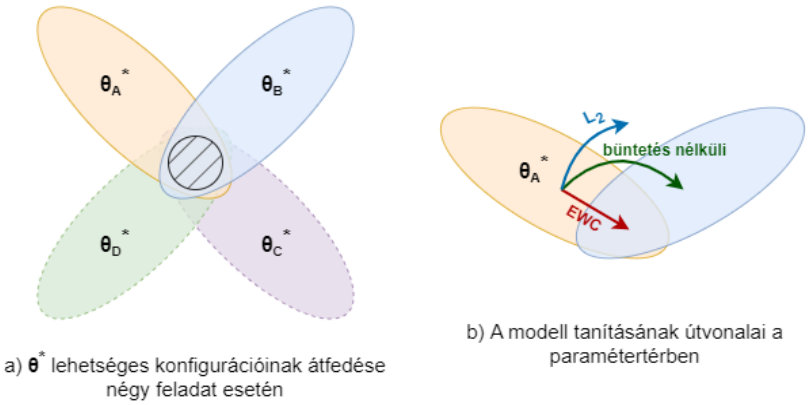
Egy mély neurális hálózatban a teljesítmény optimalizálásakor a súlyok megfelelő beállítása játszik döntő szerepet. Ez a háló $\boldsymbol{\theta}$ paramétervektora optimális értékének megtalálása. A tanítás során létrejön egy leképezés a bemenet tere és a kimeneti tér között, mely során a cél egy olyan optimális $\boldsymbol{\theta} = \boldsymbol{\theta}^*$ paraméter megtalálása, amely a legkisebb hibát eredményezi a hibafüggvényen keresztül mérve.

Azonban több optimális $\boldsymbol{\theta}$ konfiguráció is lehetséges, melyek egy megoldási teret alkotnak (a globális optimum megtalálására nincs garancia, legtöbbször lokális

optimumokat talál a gradiens alapú optimalizálás). Ezen $\boldsymbol{\theta}^*$ konfigurációk a legoptimálisabb $\boldsymbol{\theta}$ körüli térként értelmezhetők, a tanítás során elfogadottnak vélt hibaszázalékokkal. A 2. ábrán az ellipszisek ábrázolják az egyes feladatokhoz tartozó optimális $\boldsymbol{\theta}$ körüli konfigurációk terét a megengedett hibatartományokon belül, ezen ellipszisek metszete pedig az összes feladat közös megoldási terét jelenti.

A modern neurális hálók túlparaméterezése miatt azonban feltehető, hogy a B feladatnak létezik egy olyan $\boldsymbol{\theta}_B^*$ megoldása, mely igen közel áll a már korábban megtalált A feladat $\boldsymbol{\theta}_A^*$ megoldásához. Ezért egy B feladat tanulása során az EWC a paraméterek korlátozásával igyekszik megőrizni az A feladaton az azon való tanulás végén elért teljesítményt úgy, hogy a paraméterek az A feladat egy adott $\boldsymbol{\theta}_A^*$ pontja körüli alacsony hibatartományban maradjanak.

Ezen korlátozás nem lehet minden paraméternél azonos, a hatékonyság érdekében nagyobb érték beállítása szükséges azon paramétereknél, amelyek a legfontosabbak az A feladat során nyújtott teljesítmény szempontjából. Természetesen a tanítás során élünk azzal a feltételezéssel, miszerint mindig van egy átfedő metszet az összes feladat megoldási terei között. A megfogalmazott optmimumkeresési elvet szemlélteti az alábbi 2. ábra.



2. ábra. Az EWC működésének alapjai - optimumok a paramétertérben.

A 2. ábra bal oldali részén csíkozott jelöléssel látható azon paramétertartomány, amely az összes feladat közös megoldási terét jelöli, tehát ahol a modell a korábbi feladatokon katasztrofikus felejtés nélkül képes működni. A jobb oldali ábrán az látható, hogyan lehet képes egy modell a B feladat tanulása során emlékezni a már megtanult A feladatra. Ha a zöld nyíllal ábrázolt tanítási útvonalat tekintjük (a paraméterek optimalizálása során iteratívan meglépett paramétefrissítések irányított vektoroknak feleltethetőek meg a paramétertérben), akkor az A feladat megtanulása után gradiens lépéseket téve minimalizáljuk a B feladat felejtését, azonban elveszítjük az A-nál tanultakat. A kék nyíllal jelölt esetben ha minden súlyt egyforma együtthatóval

korlátozunk le, akkor a túl szigorú megszorítás miatt az A feladatra csak akkor képes emlékezni a modell, ha nem tanulja meg a B-t. Ezen esetek megoldására szolgál az EWC, amely megoldást szolgál a B feladat megtanulására az A feladat elfelejtése nélkül.

3.1.1. Az EWC megértéséhez elengedhetetlen definíciók

Ezen fejezetben említésre kerül két olyan definíció, melyek ismerete és megértése az EWC megvalósításához elengedhetetlen.

3.1. Definíció. Laplace approximáció.

A Laplace-módszer az f függvény $f(\boldsymbol{\theta})d\boldsymbol{\theta}$ integrálját közelíti az $f(\boldsymbol{\theta})$ maximumra való Gauss függvény illesztésével, majd annak térfogatszámításával.

Ehhez először megkeressük az $f(\boldsymbol{\theta})$ függvény maximumát, amely általában a legvalószínűbb pont vagy a maximum poszteriori (MAP) becslés. Ezután a függvény maximuma körül egy Gauss-eloszlást illesztünk. Ez úgy történik, hogy az $f(\boldsymbol{\theta})$ másodrendű deriváltjával, a Hesse-mátrixszal közelítjük a függvény görbületét a maximum pontjában, ami meghatározza a Gauss-eloszlás szórását. Végül a Gauss-eloszlással közelített függvény térfogatát (integrálját) számítjuk ki, ami egyszerűen a normális eloszlás integrálja. Ez a módszer különösen hasznos, mert a bonyolult integrálokat egy egyszerűbb, analitikusan kezelhető formával közelíti.

3.2. Definíció. Fisher információs mátrix.

Legyen l a likelihood függvény természetes alapú logaritmusának $\boldsymbol{\theta}$-hoz viszonyított parciális deriváltja. Ekkor, ha X egy valószínűségi változó, amelynek eloszlása $\boldsymbol{\theta}$ paraméterektől függ, és sűrűségfüggvénye $f(x, \boldsymbol{\theta})$, akkor l várható értéke $\boldsymbol{\theta}$ esetén nulla:

$$E\left[\frac{\partial}{\partial \boldsymbol{\theta}} \log f(X, \boldsymbol{\theta}) \middle| \boldsymbol{\theta}\right] = \frac{\partial}{\partial \boldsymbol{\theta}} \int_{\mathbb{R}} f(x, \boldsymbol{\theta}) dx = 0.$$

Az l varianciáját nevezzük Fisher információnak:

$$I(\boldsymbol{\theta}) = E\left[\left(\frac{\partial}{\partial \boldsymbol{\theta}} \log f(X, \boldsymbol{\theta})\right)^2 \middle| \boldsymbol{\theta}\right] = \int_{\mathbb{R}} \left(\frac{\partial}{\partial \boldsymbol{\theta}} \log f(x, \boldsymbol{\theta})\right)^2 f(x, \boldsymbol{\theta}) dx,$$

N darab paraméter esetén pedig, ahol $\boldsymbol{\theta} = [\theta_1, \theta_2, \ldots, \theta_N]^T$, a Fisher információ egy $N \times N$-es mátrix:

$$[I(\boldsymbol{\theta})]_{i,j} = E\left[\left(\frac{\partial}{\partial \boldsymbol{\theta}_i} \log f(X, \boldsymbol{\theta})\right)\left(\frac{\partial}{\partial \boldsymbol{\theta}_j} \log f(X, \boldsymbol{\theta})\right) \middle| \boldsymbol{\theta}\right],$$

melyet Fisher információs mátrixnak nevezünk.

3.1.2. Az EWC módszer

Annak meghatározásához, hogy mely súlyok a legfontosabbak egy adott feladat tanulása során, célszerű valószínűségi szempontból vizsgálni a neurális hálózatok tanítását. Ezen értelmezés mellett a paraméterek optimalizálása ekvivalens a legvalószínűbb paraméterbeállítás megtalálásával egy adott D adathalmazon. Tehát a cél a $p(\boldsymbol{\theta}|D)$ poszteriori valószínűségi eloszlásfüggvény meghatározása a D adathalmaz ismeretében.

A szükséges $p(\boldsymbol{\theta}|D)$ feltételes valószínűséget a $p(\boldsymbol{\theta})$ valószínűségből és a $p(D|\boldsymbol{\theta})$ feltételes valószínűségből kapjuk a Bayes-szabály alkalmazásával:

$$p(\boldsymbol{\theta}|D) = \frac{p(D|\boldsymbol{\theta})\ p(\boldsymbol{\theta})}{p(D)}. \tag{1}$$

Mivel egy függvény maximalizálása megegyezik annak logaritmusának maximalizálásával, ezért a fenti egyenlet átírható az alábbi alakra:

$$\log p(\boldsymbol{\theta}|D) = \log p(D|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) - \log p(D). \tag{2}$$

Egy neurális hálózatot a D adathalmazon való tanítása ekvivalens a paraméterek loglikelihood függvényének maximalizálásával:

$$\arg\max_{\boldsymbol{\theta}} \Big\{ l(\boldsymbol{\theta}) = \log p(\boldsymbol{\theta}|D) \Big\}. \tag{3}$$

Tegyük fel, hogy az adatokat két független részre osztjuk, legyenek ezek A és B (az alábbi egyenletnél először D_A és D_B jelöléssel), továbbá $D = \{A, B\}$ és B az A-t követő feladat. Ekkor a (2) egyenlet két független feladatra való átírása:

$$\log p(\boldsymbol{\theta}|D) = \log p(D_B|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}|D_A) - \log p(D_B). \tag{4}$$

Ezt tovább egyszerűsítve és formázva kapjuk az alábbi alakot:

$$\log p(\boldsymbol{\theta}|D) = \log p(B|A,\ \boldsymbol{\theta}) + \log p(\boldsymbol{\theta}|A) - \log p(B|A). \tag{5}$$

Majd a fenti egyenletet tovább alakítva kapjuk a végleges alakot:

$$\log p(\boldsymbol{\theta}|D) = \log p(B|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}|A) - \log p(B). \tag{6}$$

Ekkor $p(B|\boldsymbol{\theta})$ a B feladathoz tartozó veszteség az adott $\boldsymbol{\theta}$ paraméterkonfiguráció mellett, $p(B)$ a B feladat valószínűsége, az eredetileg A feladatra vonatkozó $p(\boldsymbol{\theta}|A)$ poszteriori eloszlás pedig most a $p(\boldsymbol{\theta}|B)$ poszterior valószínűségi eloszlásává válik a B feladatban. Ez utóbbi azt jelenti, hogy az A feladat után megszerzett ismeretek most a B feladat kezdeti feltételeinek számítanak, így ezt a tudást egy priori eloszlásként használjuk, amikor a B feladatra térünk át.

Emellett fontos megjegyezni, hogy a kapott (5) egyenlet bal oldala a paraméterek teljes adathalmazra vonatkozó valószínűségét adja meg, jobb oldala viszont csak a B feladat $\log p(B|\boldsymbol{\theta})$ veszteségfüggvényétől függ, ezért az A feladat minden információjának a $p(\boldsymbol{\theta}|A)$ eloszlásban kell lennie. Ennek a valószínűségnek kell információt tárolnia arról, hogy mely paraméterek játszottak döntő szerepet az A feladat során. Ez az EWC módszer megvalósításának legfontosabb lépése.

Azonban a $p(\boldsymbol{\theta}|A)$ poszteriori valószínűségi eloszlás nehezen kezelhető, ezért a Laplace-approximáció módszerét alkalmazzuk annak érdekében, hogy egy megfelelő közelítést találjunk a folytonos valószínűségi sűrűségfüggvényhez. Azzal a feltételezéssel élve, hogy $p(\boldsymbol{\theta}|A)$ a maximuma körül koncentrálódik (vagyis $\boldsymbol{\theta}^*_A$), ezt a valószínűségi eloszlást egy normális eloszlással közelíthetjük, melynek középértéke $\boldsymbol{\theta}^*_A$ és varianciája $[F]^{-1}$, ahol F a Fisher-információs mátrixot jelöli.

Ekkor tehát a poszterior eloszlást Laplace-approximációval Gauss-eloszlásként közelítjük, ahol $\boldsymbol{\theta}^*_A$ a maximális posteriori becslés és az F Fisher-információs mátrix inverze adja a kovarianciamátrixot.

Az EWC-ben szereplő F Fisher-mátrixnak három lényeges tulajdonsággal kell rendelkeznie: (a) a veszteségfüggvény második deriváltjának minimumával ekvivalens, (b) csakis elsőrendű deriváltakból számítható ki a nagy modellekre való tekintettel, és (c) pozitív szemidefinitnek kell lennie.

Mindezen fenti feltételeket figyelembe véve az EWC-ben levő minimalizálandó L függvény a következő (az $L(y, F_{\boldsymbol{\theta}}(x))$ veszteségfüggvényt a képletben az egyszerűbb átláthatóság végett $L(\boldsymbol{\theta})$ jelöli.):

$$L(\boldsymbol{\theta}) = L_B(\boldsymbol{\theta}) + \sum_k \frac{\lambda}{2} F_k(\boldsymbol{\theta}_k - \boldsymbol{\theta}^*_{A,k})^2,$$

ahol $L_B(\boldsymbol{\theta})$ a B feladat vesztesége, i végigfut minden paraméteren, a λ pedig a régi feladat fontosságát szabályozza az új feladathoz képest.

Egy harmadik, C feladatra való áttéréskor az EWC közel tartja a hálózati paramétereket az A és a B feladat tanult paramétereihez egyaránt, mely elérhető egy feltétel, vagy két külön feltétel alkalmazásával is. Előbbi esetben a két másodfokú feltétel összege is másodfokú lesz.

3.1.3. A kapott értékek fontossága

Ekkor felmerül a kérdés, hogy vajon hogyan kaptuk pont ezeket az értékeket? Erre a kérdésre ad választ ez az alfejezet.

Első lépésben kiszámítjuk az $l(\boldsymbol{\theta})$ másodrendű Taylor-sorfejtését $\boldsymbol{\theta}_A^*$ körül a következőképpen:

$$l(\boldsymbol{\theta}) \approx l(\boldsymbol{\theta}_A^*) + \left(\frac{\partial l(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \bigg|_{\boldsymbol{\theta}_A^*} \right) (\boldsymbol{\theta} - \boldsymbol{\theta}_A^*) + \frac{1}{2} (\boldsymbol{\theta} - \boldsymbol{\theta}_A^*)^\top \left(\frac{\partial^2 l(\boldsymbol{\theta})}{\partial^2 \boldsymbol{\theta}} \bigg|_{\boldsymbol{\theta}_A^*} \right) (\boldsymbol{\theta} - \boldsymbol{\theta}_A^*) + \dots . \quad (7)$$

Ekkor a lineáris tag elhagyható, hiszen másodfokú közelítésre van szükségünk. Így az egyenlet csak a másodrendű taggal és a konstanssal a következő:

$$l(\boldsymbol{\theta}) \approx l(\boldsymbol{\theta}_A^*) + \left(\frac{\partial l(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \bigg|_{\boldsymbol{\theta}_A^*} \right) + \frac{1}{2} (\boldsymbol{\theta} - \boldsymbol{\theta}_A^*)^\top \left(\frac{\partial^2 l(\boldsymbol{\theta})}{\partial^2 \boldsymbol{\theta}} \bigg|_{\boldsymbol{\theta}_A^*} \right) (\boldsymbol{\theta} - \boldsymbol{\theta}_A^*) + \dots . \quad (8)$$

Az egyenletek végén szereplő ... jelzés a további magasabb rendű hibatagok meglétére vonatkozik, melyek számunkra most jelentéktelenek, ezeket elhagyjuk.

Továbbá mivel a log-likelihood függvény gradiense a $\boldsymbol{\theta}_A^*$ pontban 0 – hiszen az érintő meredeksége a csúcsponton 0 –, ezért $\frac{\partial l(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \Big|_{\boldsymbol{\theta}_A^*} = 0$ miatt a következőt kapjuk:

$$l(\boldsymbol{\theta}) \approx l(\boldsymbol{\theta}_A^*) + \frac{1}{2} (\boldsymbol{\theta} - \boldsymbol{\theta}_A^*)^\top \left(\frac{\partial^2 l(\boldsymbol{\theta})}{\partial^2 \boldsymbol{\theta}} \bigg|_{\boldsymbol{\theta}_A^*} \right) (\boldsymbol{\theta} - \boldsymbol{\theta}_A^*). \quad (9)$$

Ekkor az $l(\boldsymbol{\theta}) = \log p(\boldsymbol{\theta}|D)$ helyettesítést használva az előző egyenlet A feladatra való felírása:

$$\log p(\boldsymbol{\theta}|A) = \log p(\boldsymbol{\theta}_A^*|A) + \frac{1}{2} (\boldsymbol{\theta} - \boldsymbol{\theta}_A^*)^\top \left(\frac{\partial^2 \log p(\boldsymbol{\theta}|A)}{\partial^2 \boldsymbol{\theta}} \bigg|_{\boldsymbol{\theta}_A^*} \right) (\boldsymbol{\theta} - \boldsymbol{\theta}_A^*). \quad (10)$$

A következő lépésben írjuk fel a $\left(\frac{\partial^2 \log p(\boldsymbol{\theta}|A)}{\partial^2 \boldsymbol{\theta}} \Big|_{\boldsymbol{\theta}_A^*} \right)$ tagot $- \left(\left(- \frac{\partial^2 \log p(\boldsymbol{\theta}|A)}{\partial^2 \boldsymbol{\theta}} \Big|_{\boldsymbol{\theta}_A^*} \right)^{-1} \right)^{-1}$ alakban. Ekkor ezt visszahelyettesítve az eredeti egyenletbe az alábbit kapjuk:

$$p(\boldsymbol{\theta}|A) = \epsilon \exp \left(-\frac{1}{2} (\boldsymbol{\theta} - \boldsymbol{\theta}_A^*)^\top \left(\left(- \frac{\partial^2 \log p(\boldsymbol{\theta}|A)}{\partial^2 \boldsymbol{\theta}} \bigg|_{\boldsymbol{\theta}_A^*} \right)^{-1} \right)^{-1} (\boldsymbol{\theta} - \boldsymbol{\theta}_A^*) \right),$$

ahol $\epsilon = \exp(\log p(\boldsymbol{\theta}_A^*|A))$ egy konstans. Ebből pedig már látható miként kapjuk, hogy a keresett $p(\boldsymbol{\theta}|A)$ egy $\boldsymbol{\theta}_A^*$ várható értékű és $\left(- \frac{\partial^2 \log p(\boldsymbol{\theta}|A)}{\partial^2 \boldsymbol{\theta}} \Big|_{\boldsymbol{\theta}_A^*} \right)^{-1}$ szórásnégyzetű normális eloszlással közelíthető:

$$p(\boldsymbol{\theta}|A) \sim \mathcal{N} \left(\boldsymbol{\theta}_A^*, \left(- \frac{\partial^2 \log p(\boldsymbol{\theta}|A)}{\partial^2 \boldsymbol{\theta}} \bigg|_{\boldsymbol{\theta}_A^*} \right)^{-1} \right). \quad (11)$$

3.1.4. A Fisher mátrix fontossága

Ez a fejezet arra ad választ, hogyan kapcsolódik a Fisher-információs mátrix a feladatokhoz tartozó paraméterek fontosságához.

Egy hálózatra akkor mondjuk, hogy megtanulta az adott feladatot, ha sikerült minimalizálni a hibafüggvényt a tanító adatokon, vagyis a hálózat aktuális paraméterei a hibafüggvény-felület egy minimumának feleltethetők meg. A felület görbülete megadja a hálózat érzékenységét a $\boldsymbol{\theta}^*$ optimumra vonatkozóan, ez az érzékenység pedig meghatározható azt az irányt vizsgálva, amerre $\boldsymbol{\theta}^*$ változik. A görbület fordítottan arányos $\boldsymbol{\theta}^*$ változásával, melyből adódóan minél nagyobb a görbület, annál nagyobb változás következhet be a hibafüggvény értékében a $\boldsymbol{\theta}^*$-tól való elmozduláskor.

Az említett görbületet matematikailag a másodrendű derivált, a Hesse mátrix határozza meg. Ezen esetben a Fisher-információs mátrix a poszterior log-likelihood függvény másodrendű deriváltjaként jön képbe. Ebben található mely paraméterek fontosak egy adott feladat során. Ha egy paraméterhez tartozó elem a Fisher-információs mátrixban nagy, akkor az azt jelenti, hogy az a paraméter fontos a feladatban elért teljesítmény megőrzése szempontjából.

Induljunk ki $p(\boldsymbol{\theta}|A)$ becsült normális eloszlásának varianciájából. Ahogy azt már az előző fejezetekben is láttuk, adott $\boldsymbol{\theta}_A^*$ esetén a $\log p(\boldsymbol{\theta}|A)$ kifejezés a $p(\boldsymbol{\theta}|A)$ poszterior valószínűségi sűrűségfüggvény logaritmusa, azaz a log-likelihood értéke. Más szóval a valószínűségi sűrűségfüggvény értékét adja meg az adott $\boldsymbol{\theta}_A^*$ pontban. Ez pedig a Fisher-információs mátrix inverzét jelenti:

$$F_A = \mathbb{E}\left[-\frac{\partial^2 \log p(\boldsymbol{\theta}|A)}{\partial^2 \boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}_A^*}\right].$$

Fontos megjegyezni, hogy F_A-t az (1) felhasználásával kaptuk, azzal a feltételezéssel, miszerint $p(\boldsymbol{\theta})$ és $p(A)$ konstans. Innen tudjuk a már korábban meghatározott $p(\boldsymbol{\theta}|A) \sim \mathcal{N}(\boldsymbol{\theta}_A^*, F_A^{-1})$ értéket (11).

Azonban az előző fejezet (9)-es lépésében történő másodrendű deriváltat igénylő számítással ellentétben a Fisher-információs mátrix kiszámítható az elsőrendű deriváltakból is a következőképpen:

$$F_A = \mathbb{E}\left[-\frac{\partial^2 \log p(\boldsymbol{\theta}|A)}{\partial^2 \boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}_A^*}\right] = \left[\left(\left(\frac{\partial \log p(\boldsymbol{\theta}|A)}{\partial \boldsymbol{\theta}}\right)\left(\frac{\partial \log p(\boldsymbol{\theta}|A)}{\partial \boldsymbol{\theta}}\right)^\top\right)\Big|_{\boldsymbol{\theta}_A^*}\right]. \quad (12)$$

Ezután visszahelyettesítjük a (10)-es egyenletet a (6)-osba és a következőt kapjuk:

$$\begin{aligned}\log p(\boldsymbol{\theta}|D) &= \log p(B|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}|A) - \log p(B) \\ &= \log p(B|\boldsymbol{\theta}) + \frac{\lambda}{2}\left(\boldsymbol{\theta} - \boldsymbol{\theta}^*_A\right)^\top \left(\frac{\partial^2 \log p(\boldsymbol{\theta}|A)}{\partial^2 \boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}^*_A}\right)\left(\boldsymbol{\theta} - \boldsymbol{\theta}^*_A\right) + \epsilon'. \quad (13)\end{aligned}$$

Itt ϵ' jelöli az összes felmerülő konstanst, λ pedig egy olyan hiperparaméter, melyet azért vezetünk be, hogy biztosítsuk a B feladat megtanulását és az A feladatban tanultak megőrzését egyaránt.

Ekkor a fenti egyenlet tovább egyszerűsítése (12) segítségével:

$$\begin{aligned}\log p(\boldsymbol{\theta}|D) &= \log p(B|\boldsymbol{\theta}) + \frac{\lambda}{2}\left(\boldsymbol{\theta} - \boldsymbol{\theta}^*_A\right)^\top \left(\frac{\partial^2 \log p(\boldsymbol{\theta}|A)}{\partial^2 \boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}^*_A}\right)\left(\boldsymbol{\theta} - \boldsymbol{\theta}^*_A\right) + \epsilon' \\ &= \log p(B|\boldsymbol{\theta}) - \frac{\lambda}{2}\left(\boldsymbol{\theta} - \boldsymbol{\theta}^*_A\right)^\top F_A\left(\boldsymbol{\theta} - \boldsymbol{\theta}^*_A\right) + \epsilon'. \quad (14)\end{aligned}$$

Ezt tovább egyszerűsítve pedig megkapjuk a végleges alakot:

$$l(\boldsymbol{\theta}) = l_B(\boldsymbol{\theta}) - \frac{\lambda}{2}\left(\boldsymbol{\theta} - \boldsymbol{\theta}^*_A\right)^\top F_A\left(\boldsymbol{\theta} - \boldsymbol{\theta}^*_A\right) + \epsilon', \quad (15)$$

ahol $l(\boldsymbol{\theta})$ jelöli az általános veszteséget, $l_B(\boldsymbol{\theta})$ a B feladatra vonatkozó veszteséget, $\left(-\frac{\lambda}{2}\left(\boldsymbol{\theta} - \boldsymbol{\theta}^*_A\right)^\top F_A\left(\boldsymbol{\theta} - \boldsymbol{\theta}^*_A\right)\right)$ pedig a súlyokat szabályozza.

Ezáltal látható miként játszik a Fisher-információs mátrix kulcsszerepet a hálózatok paraméterei fontosságának meghatározásában.

3.1.5. Az EWC hatékonysága

A [Kirkpatrick et al., 2017] cikkben a szerzők az általuk bemutatott módszert sűrűn kapcsolt rétegeket tartalmazó előrecsatolt neurális háló tanításával értékelik ki képi adaton, az MNIST kézzel írott számjegyek adathalmazán. A kíséletben az egyes feladatok a tanítóképek permutálásával keletkeznek, minden feladat egy egyedi véletlenszerű permutációval van társítva. A kísérleti eredményeik azt mutatják, hogy ebben az esetben az EWC módszerrel tanított modell sikeresen elkerüli a katasztrófikus felejtést, míg egy standard módon osztályozásra tanított háló esetén korábbi feladatok teljesítménye gyorsan romlik az új feladatokra való tanítás során.

Az eredmények alapján az EWC ígéretes megközelítésnek bizonyul a mély neurális hálózatokban történő hosszú távú tanulás megvalósítására, de későbbi cikkek eredményei, pédául [Buzzega et al., 2020] azt mutatják, hogy komplexebb képi adathalmazokon, nehezebb osztályozási feladatokkal még az EWC módszer sem segít konvolúciós háló tanítása során a katazstrofikus felejtés elkerülésére. Ugyanezt támasztják alá a szakdolgozatban bemutatott kísérletek.

3.2. Visszajátszás (ER)

A visszajátszás, avagy más néven Experience Replay [Chaudhry et al., 2019] egy másik kulcsfontosságú módszer a hosszú távú tanulás területén, amely az emberi agyban megfigyelhető emlékezés-mechanizmusok mintájára a tanulási folyamat során korábban látott adatokat külön memóriában tárolva újra felhasználja azokat, úgymond emlékezteti a modellt az korábbi feladatokra. Az előbbi fejezetben tanulmányozott EWC-hez hasonlóan ez a módszer is a katasztrofikus felejtés problémájának megoldását hivatott megoldani, azonban lényegesen más megközelítést alkalmaz ennek megvalósításához.

A visszajátszás működésének alapja, hogy az aktuális feladatok példáit és az epizodikus memóriában tárolt példákat együttesen használja fel a modell tanításához, ezzel javítva a pontosságot és csökkentve a katasztrofikus felejtést. Epizodikus memória alatt egy olyan memóriát értünk, amely a tanult adatokhoz kapcsolódó konkrét emlékeket és tapasztalatokat tartalmazza. Ezen memória teszi lehetővé a visszajátszás során, hogy a rendszer tárolja és újrahasznosítsa a korábban megtanult adatpontokat.

3.3. Definíció. Adott egy feladathoz tartozó $D^t = \{(\boldsymbol{x}_i^t, y_i^t) \,|\, i = 1, 2, \ldots, |D^t|\}$ adathalmaz. Ekkor a visszajátszás során egy rögzített K_M kapacitású M memória tárolja ezeknek az adatpontoknak egy részét, amelyeket a tanítási folyamat során újra felhasználunk. A memória minden megtanult feladat adataiból tárolhat elemeket, az aktuálisan tanult feladat során dől el, hogy melyek kerülnek bele.

Az egyszerűsége miatt egyik leggyakrabban használt memória az úgynevezett reservoir memória, amelynek a tartalma a reservoir mintavételezés [Vitter, 1985] alapján kerül kiválasztásra. A memória használatának lépései:

1. *Memória frissítése*
   Minden $(\boldsymbol{x}_i^t, y_i^t)$ adatpontot hozzáadunk M-hez, egészen addig, amíg a memória kapacitása engedi. A K_M korlát elérése után az új adatpont egyre kisebb valószínűséggel kerülhet be a memóriába. A feladatokat adatstreamnek tekintve a modell által a tanítás során látott összes adatpontok számát jelölje S. Csak akkor kerül be az új adatpont, ha az $\{1, 2, \ldots, S\}$ halmazból véletlenszerűen sorsolt index kisebb, mint a memória kapacitása, ekkor memóriában ezen indexű kiválasztott helyen lévő adatpont törlésre kerül, azt cseréli az újonnan bekerülő.

$$\begin{cases} M \leftarrow M \cup \{(\boldsymbol{x}_i^t, y_i^t)\}, \text{ ha } |M| < K_M \\ M[j] \leftarrow (\boldsymbol{x}_i^t, y_i^t), \text{ ha } j < K_M, \text{ ahol } j \in \{0, 1, \ldots, S\} \text{ véletlenszerű.} \end{cases}$$

2. *Mintavételezés a memóriából*

   Az aktuális modell frissítése érdekében egy mini-batch méretű B_M adatponthalmazt véletlenszerűen mintavételezünk M-ből: $B_M \sim M$.

3. *Modell-frissítés*

   A modell paramétereinek frissítése során az adott feladat tanulásához a feladat adathalmazából származó $B \sim D^t$ mini-batchen számolt veszteség mellett figyelembe vesszük a memóriából származó mini-batchen számolt veszteséget is, ezeknek az összegét tekintjük. Az így összesített veszteség alapján a modell paramétereit gradiens alapú optimalizálással frissítjük.

$$L(\boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim B} L(\boldsymbol{y}, F_{\boldsymbol{\theta}}(\boldsymbol{x})) + \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim B_M} L(\boldsymbol{y}, F_{\boldsymbol{\theta}}(\boldsymbol{x})).$$

4. Kísérletek

A kísérletek során mély mesterséges neurális hálók esetében öt különböző tanítási módszert hasonlítottunk össze, képi adatokon tanítva őket. Először egy standard tanítást végeztünk a teljes adathalmazon, ez egy elérhető felső korlátként szolgál a szekvenciális hosszú távú tanulás vizsgálatához. Hasonlóan, alsó korlátként egy javítási módszerek nélküli hosszú távú tanítást végeztünk el, majd az EWC és/vagy a visszajátszás módszerekkel történő hosszú távú tanulást hajtottuk végre és elemeztük.

A hosszú távú tanulásra vonatkozó kísérletek során a 2.4 fejezetben említett inkrementális feladattanulást valósítottuk meg a CIFAR-10 adathalmazon [Krizhevsky et al., 2009]. Ehhez a tanítási és tesztelési adathalmazt diszjunkt részhalmazokra, egész pontosan öt feladatra osztottuk, melyek mindegyike két-két osztályhoz tartozó bemeneti adatot tartalmaz feladatonként. Ugyanazt a modellt szekvenciálisan tanítottuk mindegyik feladaton, végül pedig a végső modellt az összes feladat teszt adathalmazán külön kiértékeltük és a kapott pontosságokat kiátlagoltuk, az így kapott végső átlagos teszt-pontossággal jellemeztük a teljesítményét.

A kísérletek során használt kódok publikusan elérhetőek a `https://github.com/vasslea/Bsc_Szakdolgozat` oldalon. Ezen kódok használatával minden kísérleti eredmény reprodukálható. A kísérleteket Colab környezetben futtattam, egy Tesla T4 GPU gyorsítását használva.

4.1. A modelltanítás részletei

Adathalmaz. Az általunk használt adathalmaz a CIFAR-10 [Krizhevsky et al., 2009], mely 60.000 képet tartalmaz 10 különböző osztályba sorolva. Az adatok egyenletes eloszlása érdekében minden osztályhoz 6.000 kép tartozik. Minden kép 32×32-es méretű, és 3 színcsatornával rendelkezik. A tanító adathalmaz 50.000 képet tartalmaz, míg a tesztelésre szánt adatrész 10.000 képet. A képeket mindig normalizáljuk a tanítóhalmazon kiszámolt színcsatornánkénti átlaggal és szórással.

Augmentáció. Az augmentáció egy olyan kritikus technika a gépi tanulásban, melynek lényege, hogy a bementi adatokra különbőző transzformációkat alkalmazunk. Ezáltal növeljük a tanító adatok sokféleségét anélkül, hogy újat kellene gyűjtenünk. A tanítás során az augmentáció által arra kényszerül a modell, hogy az alkalmazott transzformációkra invariánsan mindig az eredei kép osztályát jósolja. A kísérletek során a képeken alkalmazott transzformációk a véletlenszerű kivágás (*RandomCrop*) és véletlenszerű vízszintes tükrözés (*RandomHorizontalFlip*) voltak.

Hiperparaméterek. A kísérletek során többféle paraméterrel tanítottuk a modellt a legmagasabb tesztpontosság elérése érdekében. A legnagyobb pontosságot a következő

paraméter-megválasztásokkal sikerült elérni: a mini-batch mérete 128, a sztochasztikus gradiens ereszkedés (SGD) optimalizáló tanulási rátája 0.1, a momentum 0.9, a súlycsökkentés (weight decay) pedig $1e-4$ mértékű.

Hálóarchitektúra. A kísérlet során egy ResNet modellt használunk. A háló részletes felépítése a 1. és 2. táblázatokban látható.

ResNet-18 architektúra - $3 \times 32 \times 32$ méretű bemenetre
Konvolúció [64 3x3-as filter], 1-es stride, 1-es padding, bias nélkül
BatchNorm [64]
ReLU
1. Réteg
BasicBlock [64 3x3-as filter, 1-es stride]
BasicBlock [64 3x3-as filter, 1-es stride]
2. Réteg
BasicBlock [128 3x3-as filter, 2-es stride]
BasicBlock [128 3x3-as filter, 1-es stride]
3. Réteg
BasicBlock [256 3x3-as filter, 2-es stride]
BasicBlock [256 3x3-as filter, 1-es stride]
4. Réteg
BasicBlock [512 3x3-as filter, 2-es stride]
BasicBlock [512 3x3-as filter, 1-es stride]
AdaptiveAvgPool2d((1, 1))
Linear [512, C]

1. táblázat. A használt ResNet konvolúciós neurális háló architektúrája.

BasicBlock architektúra
Konvolúció [3x3-as filterek, stride, 1-es padding]
BatchNorm [ ]
ReLU
Konvolúció [3x3-as filterek, stride, 1-es padding]
BatchNorm [ ]
Shortcut
ReLU

2. táblázat. A BasicBlock rétegei.

A modell BasicBlock struktúrákból épül főként, azok mindegyikben két konvolúciós réteggel. A kezdeti konvolúciós réteg 64 filtert alkalmaz 3×3-as kernel mérettel, majd ezt BatchNorm normalizálás és ReLU aktivációs függvény követi. Az első réteg két BasicBlock rétegből áll, mindkettő 64 filtert használ, míg a következő három réteg ugyanígy BasicBlock rétegek sorra 128, 256, illetve 512 filterrel. Utóbbi kettő esetében az első BasicBlock során 2-es stride-ot használunk a térbeli méret csökkentése érdekében. Legvégül egy Adaptive Average Pooling réteg és egy sűrűn kapcsolt lineáris réteg van Softmax aktivációval. A leírt hálóarchitektúra a ResNet-18 modellnek felel meg, az első konvolúciós réteg filter-mérete 7x7-ről 3x3-ra van csökkentve szokásosan a CIFAR-10 adathalmazhoz való adaptálás érdekében, a szakirodalomban ebben a formában használják legtöbben [Buzzega et al., 2020].

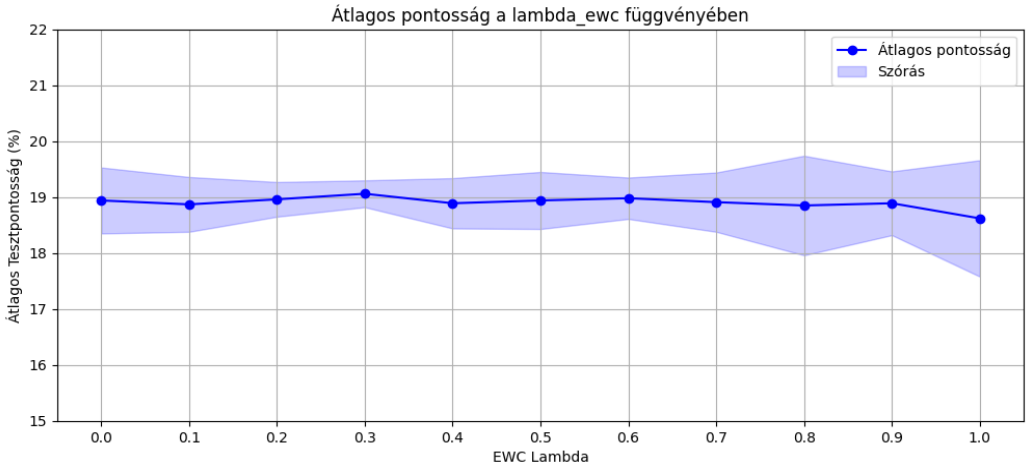
Tanítás. A szakdolgozatban használt három hosszú távú tanulási technika egyenértékű összehasonlítása végett a hálót mindegyik esetben a sztochasztikus gradiens ereszkedés optimalizáló algoritmussal tanítjuk. Emellett a kísérletek gyorsabb futtatása érdekében mindhárom esetben csak 20 epochig tanítunk feladatonként, ezalatt is midnen feladatot megtanul. 50 vagy akár 100 epochnyi tanítással esetleg jobb eremények is elérhetőek lennének. A standard tanítás 50 epochig tartott ehhez képest, hogy a teljes adathalmazon megfelelően betanuljon a modell.

4.2. Eredmények

Ezen fejezetben kerülnek kiértékelésre a dolgozatban bevezetett hosszú távú tanulási módszerek.

4.2.1. EWC

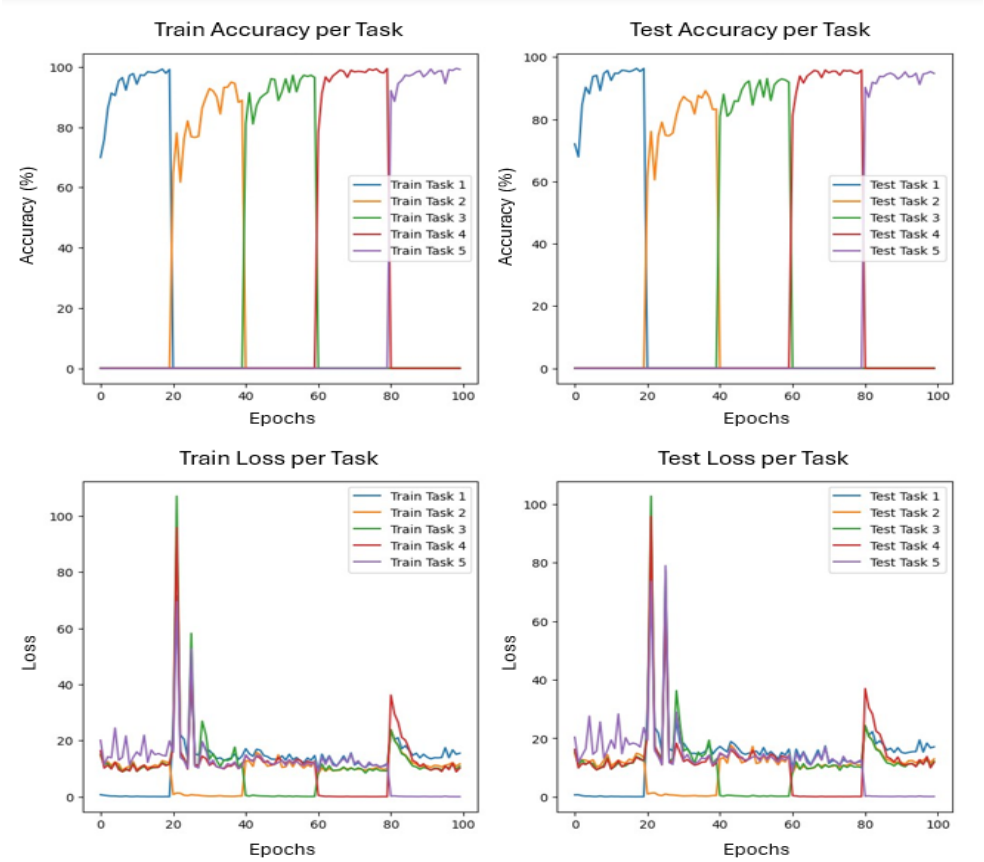
A 3.1 fejezet alapján az EWC módszer megvalósításakor úgy minimalizáljuk a katasztrofikus felejtést, hogy büntetjük a fontos modellparaméterek változását. Az implementáció során az *EWC osztály* kiszámítja és tárolja a modellparamétereknek feladathoz tartozó adthalmazon vett Fisher információs mátrixát és az előző feladaton megtanult jó paraméter-beállítást minden feladat után. Ezek felhasználásával a tanítás során az EWC hibafüggvényét számolja az új feladatokra, hogy megakadályozza a korábban megtanult fontos paraméterek változását. Az EWC hibafüggvény számolásában alkalmazunk egy *ewc_lambda* paramétert a regularizációs hibafüggvénytag fontosságának beállítására. Pontosabban, ez a paraméter egy súly, amely megadja mekkora mértékben szeretnénk büntetni a korábban megtanult fontos paraméterek változását az új feladatok tanítása során.



3. ábra. Átlagos tesztpontosság az EWC lambda függvényében.

A 3. ábrán a modell által elért végső átlagos tesztpontosság és az egyes feladatok tesztpontosságának ettől való eltéréséből származó szórás látható $[0, 1]$ közti *ewc_lambda* értékekre. Ugyan a mért értékek között a különbségek minimálisak, a legjobb pontosságot, illetve a legkisebb szórást az $ewc\_lambda = 0.3$ beállítással értük el, így a további tesztelések során ezt az értéket tartjuk meg.

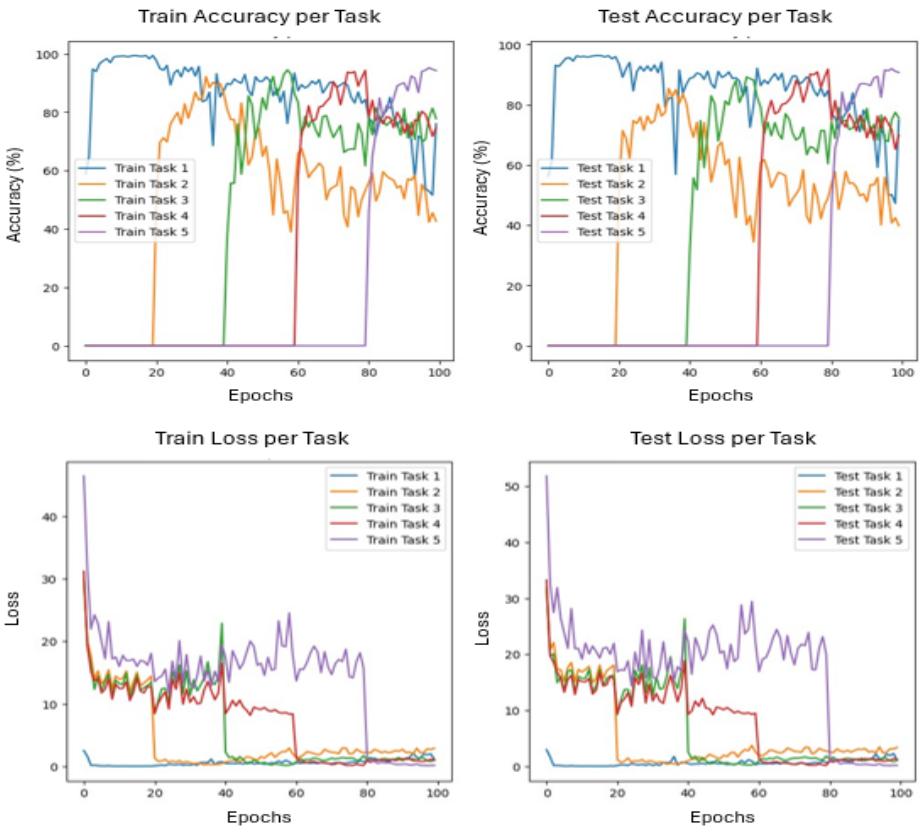
Az EWC módszerrel a telsej tanítás során mért eredményeket az 4. ábra mutatja.



4. ábra. Az EWC módszerrel elért eredmények.

Az ábrán az látható, hogy a teljes tanulási folyamat során feladatonként különkülön vizsgáljuk a tanítási (train) és tesztelési (test) pontosságot és a hibafüggvényértéket. Az egyes görbék az egyes feladatokon mért teljesítményt reprezentálják a tanítási epochok függvényében. Egy epoch alatt annyi mini-batchen történő sztochasztikus gradiens ereszkedés iterációt értünk, ahány alatt a mini-batchek a teljes adathalmazt lefedik egyszer. Az ábrán megfigyelhető, hogy a tanítási és a tesztelési pontosság az első feladat tanítása során gyorsan növekszik. Azonban az új feladatok bevezetésekor a korábbi feladatok teljesítménye hirtelen és drasztikusan lecsökken, majd csak az aktuális feladat tanítási pontossága növekszik. Mivel mindig csak az aktuális feladaton magas a tanítási és a tesztelési pontosság, ez arra utal, hogy a modell a korábbi feladatokat elfelejti, ami a katasztrofikus felejtés pontos megnyilvánulása. Az EWC módszer célja az lenne, hogy megőrizze a korábban szerzett ismereteket, azonban az ábrán látható, miszerint ezen a nehéz feladaton ez nem igazán sikerül. További hasonló eredmények a szakirodalomból [Buzzega et al., 2020] alátámasztják, hogy ez a módszer bonyolultabb feladatokon kevés sikerrel alkalmazható.

4.2.2. Visszajátszás



5. ábra. Az ER módszerrel elért eredmények.

Ezen módszer lényege a 3.2 fejezet alapján az, hogy a hálózat nem csak az aktuális feladathoz tartozó adatok alapján frissíti a súlyait, hanem egy memóriában tárolt korábbi minták alapján is, így segítve a modellnek a korábbi feladatokból szerzett ismeretek megtartását. Az implementációban a *ReplayBuffer* osztályban tároljuk a kiválasztott kép-címke párokat, melyek a reservoir mintavételezés alapján kerülnek hozzáadásra, a 3.2. részben leírt módon. Általában igaz az, hogy minél nagyobb a memória mérete, annál nagyobb pontosságot lehet elérni a hosszú távú tanulás során visszajátszással.

A 5. ábrán a visszajátszás módszerrel történő tanítás eredményei láthatóak. Az EWC implementálásával elért eredményekkel ellentétben ebben az esetben az vehető észre, hogy ugyan az új feladatok bevezetésekor a korábbi feladatok teljesítménye csökken, azonban egyáltalán nem drasztikusan. Ez azt jelenti, hogy a modell emlékszik a korábban szerzett ismeretekre is, amikor egy új feladaton tanul. Az alsó két ábrán az látszik, hogy csak a még nem látott feladatokon számolt veszteség magas, hiszen ezeket nem is tanulta még a modell. Az új feladatok bevezetésekor az aktuális feladat vesztesége is csökken, míg a korábbi feladatok alacsony vesztesége szintén megmarad.

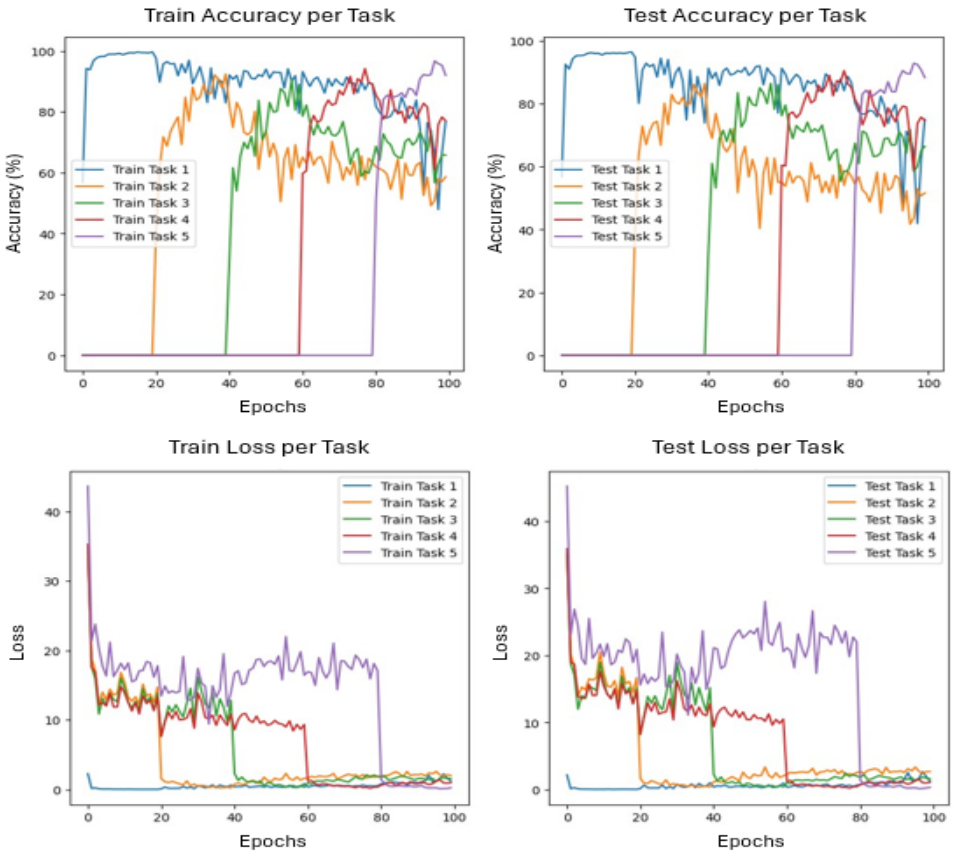
Ezek az ábrák jól szemléltetik, hogy a visszajátszás módszere lehetővé teszi a modell számára, hogy a korábbi feladatok során szerzett ismereteket részben megőrizze, miközben az új feladatokon is hatékonyan tanul. Ezáltal a katasztrofikus felejtés mértéke jelentősen csökken, azonban nem szűnik meg teljesen. Az emberi agyban megfigyelt hasonló emlékezési folyamatok mintájára a visszajátszás sikeresen "emlékezteti a modellt" a korábbi adatokon elvárt kimentek prediktálására.

4.2.3. EWC visszajátszással kombinálva

A jobb tesztpontosság elérése reményében kombináltuk a két módszert: a modellt három hibafüggvénytag egyidejű minimalizálásával tanítottuk. A három hibafüggvénytag az aktuális feladat tanulására, az EWC regularizáció használatára, illetve a memóriából történő visszajátszás alkalmazására vonatkozó megfelelő tagok.

A 6. ábra a leírt tanítás során mért eredményeket mutatja.

Az EWC és a visszajátszás ötvözésével kapott eredmények minimális eltérést mutatnak a korábbi, sima visszajátszással kapott eredményektől. Az ábrán kivehető, hogy a tanítás során a modell a visszajátszás miatt ugyanúgy képes megőrizni a teljesítményét korábbi feladatokon, mint egyszerű visszajátszással is. Az EWC használata ugyan nem eredményez szemmel láthatóan nagy javítást, de az eredmények összesítéséből kiderül, hogy végülis $1-2\%$ pontosságbeli jobb eredmény érhető el a két módszer kombinációjával. Minél több feladaton tanítjuk a modellt, annál kevésbé emlékszik a korábban megtanultakra, elképzelhető, hogy több feladat tanulásakor



6. ábra. Az EWC és ER módszerekkel elért eredmények.

már jóval jelentősebb lenne a módszerek kombinálásának a hatása.

4.2.4. A módszerek összehasonlítása

A módszerek konzisztens összehasonlítása végett ebben a részben minden módszer esetén az utolsó (5.) feldat tanítása után értékeljük ki a tesztpontosságot az összes feladaton, majd ezen értékek átlagát vesszük. Tehát ekkor azt vizsgáljuk, hogy az adott módszer az utolsó feladat tanulásának befejezése után mennyire jól tudja megoldani az összes tanult feladatot.

Az elvégzett kísérletek eredményeinek összesítését tartalmazza a 3. táblázat. A Standard jelölés a teljes adathalmazon történő standard tanítást jelöli, a CL a hagyományos hosszú távú tanulást, EWC és ER módszer használata nélkül, az EWC + ER pedig a két módszer kombinációját jelöli. A Standard, CL és Az EWC tanítás során nem használunk memóriát, ezért ott 0 a memória mérete. Az ER, illetve EWC + ER esetén három különböző memóriamérettel is elvégezzük a tanítást.

Ahogy a táblázat ereményei mutatják, az EWC és az ER kombinációjával sikerült minden memóriaméret esetében javítani az átlagos tesztpontosságon a sima visszajátszás módszerének alkalmazásához képest, bár memóriaméret növelése sokkal

Memóriaméret	**Tanítási módszer**	**Cifar-10 tesztpontosság** 5 feladat, 2-2 osztály
0	Standard	89.36
	CL	18.37
	EWC	19.06
200	ER	39.81
	EWC + ER	40.25
500	ER	51.45
	EWC + ER	52.12
5120	ER	70.94
	EWC + ER	72.53

3. táblázat. A kísérletek során elért végső átlagos tesztpontosság a különböző módszerekkel.

jelentősebb javulást eredményez ehhez képest. Ezáltal ezek az eredmények azt bizonyítják, hogy az EWC és a visszajátszás kombinációja hatékonyan segíti a korábban szerzett ismeretek megőrzését, különösen akkor, ha nagyobb memóriakapacitás áll rendelkezésre.

5. Konklúziók

A dolgozatban bemutattam az általános fogalmakat a hosszú távú tanulás módszertani alapjairól, illetve részletesen ismertettem két módszert, az elasztikus súlykonszolidációt és a visszajátszást, továbbá kísérleteket végeztem a módszerek hatékonyságának tesztelése kapcsán.

A kísérletek során kapott eredmények alapján az EWC és a visszajátszás kombinációja adta a legnagyobb átlagos tesztpontosságot. A kapott eredmények ugyan bizonyítják a visszajátszás alapú módszerek hatékony működését, azonban további kísérletekre lehet szükség a még magasabb tesztpontosság és még kisebb mértékű katasztrofikus felejtés eléréséhez. A jövőbeli kutatások folyamán érdemes lehet a következő lépéseket megfontolni:

1. *A keresztentrópia veszteségfüggvény és az utolsó osztályozási réteg lecserélése.* Érdemes lehet a softmax aktiváció és a keresztentrópia veszteség lecserélése olyan módszerre való cseréje, amely jobban kezeli a hosszú távú tanulás sajátos kihívásait, pédául energia-alapú modell tanítása lehet alternatíva [Li et al., 2022].

2. *Önfelügyelt tanulás alkalmazása.* Eredményes lehet olyan önfelügyelt tanulási technikák bevezetése, melyek növelik a modell robosztusságát és általánosítási képességét, címkézetlen adatokból való tanulás mellett.

3. *Modellek összehangolása és összevonása.* Ha az egyes feladatokon külön modelleket tanítunk és ezeket minden feladat után egy közös modellé vonjuk össze, akkor a hosszú távú tanulás célját elérhetjük, érdemes a modelleket összevonás előtt összehangolni, a különbségek minimalizálása végett [Daheim et al., 2024].

Ezek segítségével a jövőbeli kísérletek során tovább javíthatjuk a hosszú távú tanulási rendszerek teljesítményét és megbízhatóságát.

Irodalomjegyzék

Abhishek Aich. Elastic weight consolidation (ewc): Nuts and bolts. *arXiv preprint arXiv:2105.04093*, 2021.

Rahaf Aljundi. Continual learning in neural networks. *arXiv preprint arXiv:1910.02718*, 2019.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

Nico Daheim, Thomas Möllenhoff, Edoardo Ponti, Iryna Gurevych, and Mohammad Emtiyaz Khan. Model merging by uncertainty-based gradient matching. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=D7KJmfEDQP`.

Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *International Conference on Artificial Intelligence and Statistics*, pages 3762–3773. PMLR, 2020.

Raia Hadsell, Dushyant Rao, Andrei A Rusu, and Razvan Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in cognitive sciences*, 24(12):1028–1040, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Yen-Chang Hsu, Yen-Cheng Liu, Anita Ramasamy, and Zsolt Kira. Re-evaluating continual learning scenarios: A categorization and case for strong baselines. *arXiv preprint arXiv:1810.12488*, 2018.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Shuang Li, Yilun Du, Gido van de Ven, and Igor Mordatch. Energy-based models for continual learning. In *Conference on lifelong learning agents*, pages 1–22. PMLR, 2022.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71, 2019.

Haoxuan Qu, Hossein Rahmani, Li Xu, Bryan Williams, and Jun Liu. Recent advances of continual learning in computer vision: An overview. *arXiv preprint arXiv:2109.11369*, 2021.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? *Advances in neural information processing systems*, 31, 2018.

Gido M Van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

Gido M Van de Ven, Tinne Tuytelaars, and Andreas S Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4(12):1185–1197, 2022.

Jeffrey S Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985.

Mitchell Wortsman, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi. Supermasks in superposition. *Advances in Neural Information Processing Systems*, 33:15173–15184, 2020.

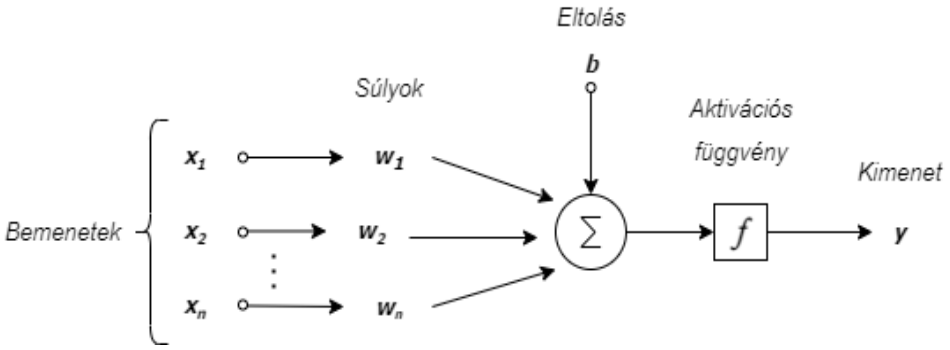
Függelék

A neurális hálózatok felépítése

A mesterséges neuronhálózatok megalkotásának motivációja az emberi agy hatékony működésének matematikai modellezése volt. Azonban az agyban lévő idegsejtek hálózatának, vagyis a biológiai neuronhálóknak a szerkezete különös bonyolultsággal rendelkezik. Ezért ugyan kisebb mérettel rendelkező, de a biológiai neuronhálók működéséhez hasonló mesterséges hálózatokat próbálunk készíteni.

I. A neuronok működése

A mesterséges neuronhálózatok alap építőeleme egy biológiai idegsejt leegyszerűsített modellje, a mesterséges neuron. Egy ilyen neuron alatt egy apró feldolgozóegységet értünk, mely bizonyos $x_1, x_2, \ldots, x_n$ bemeneti értékekkel redelkezik, majd ezen bemeneti értékek függvényében kiad egy y kimenetet. Maga a neuron egy matematikai leképezést valósít meg. Minden $x_1, x_2, \ldots, x_n$ bemeneti értékhez tartozik egy $w_1, w_2, \ldots, w_n$ súly, amely meghatározza az adott neuron "érdeklődését" a bemenet iránt. A bemenetek súlyozott összegének kiszámítása, majd az összeghez egy paraméterként kapott b eltolás-érték hozzáadása után egy aktivációs függvénynek nevezett transzformáció hajtódik végre. Egy mesterséges neuron működését pontosan szemlélteti a 7. ábra.



7. ábra. A mesterséges neuron

I.1. Definíció. Mesterséges neuron.

Ha $\boldsymbol{x} = (x_1, \cdots, x_n) \in \mathbb{R}^n, \boldsymbol{w} = (w_1, \cdots, w_n) \in \mathbb{R}^n, n \in \mathbb{N}$ és $b \in \mathbb{R}$, akkor egy n-dimenziós vektor-bemenet esetén egy neuron az alábbi $f_{\boldsymbol{\theta}} : \mathbb{R}^n \to \mathbb{R}$ paraméterezett leképezést valósítja meg:

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) \mapsto \sigma(\sum_{i=1}^{n} x_i w_i + b),$$

ahol $\boldsymbol{\theta}$ paramétervektor a neuron összes paraméterét tartalmazza, $\boldsymbol{\theta} = [\boldsymbol{w}, b]$.

I.2. Definíció. Aktivációs függvény.

A fentebb említett $\sigma : \mathbb{R} \mapsto \mathbb{R}$ függvényt nevezzük aktivációs függvénynek. Ez egy olyan matematikai függvényt foglal magába, mely egy neurális hálózatban szereplő neuron kimenetele előtt aktiválódik. Az elnevezés utal a biológiai analógiára, mely azt jelenti, hogy a függvény az adott neuron bemeneteit figyeli és amint ezen értékeknek a súlyozott összege elér egy megadott küszöböt, akkor a neuron aktiválódik. Egy egyszerű választás aktivációs függvénynek például az egységugrás lépcsős függvény:

$$\sigma(x) = \begin{cases} 0 & \text{ha } x < 0 \\ 1 & \text{ha } x \geq 0 \end{cases}$$

Ezen esetben a fenti definíció alkalmazásával n darab input érkezik, melyeknek ha a súlyozott összege eléri a $-b$ küszöbértéket, akkor a neuron aktiválódik, vagyis 1 értékkel tér vissza. Különben inaktív marad.

A neurális hálók hatékony működése érdekében a fent említett lépcsős függvény helyett a gradiens ereszkedés módszer tanítására való alkalmazásakor tipikusan folytonos és differenciálható aktivációs függvényeket használunk.
Ezek közül a leggyakrabban használt aktivációs függvények:

- **Sigmoid:**
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$
Ekkor a bemeneti érték 0 és 1 közé szorul, melyet valószínűségként is tudunk értelmezni. Emellett differenciálható, viszont hibája, hogy a gradiens elveszhet nagyon nagy vagy nagyon kicsi bemenetek esetén.

- **Tanh:**
$$\sigma(x) = \frac{2}{1 + e^{-2x}} - 1$$
Ezen esetben a bemeneti érték -1 és 1 közé esik, azonban az előző esethez hasonlóan nagyon kicsi vagy nagyon nagy bemenetek esetén a gradiens itt is elveszhet.

- **ReLU: (Rectified Linear Units)**
$$\sigma(x) = \begin{cases} 0 & \text{ha } x < 0 \\ x & \text{ha } x \geq 0 \end{cases}$$
Ennek fő tulajdonsága, hogy a negatív értékeket 0-ra állítja, a pozitívakat pedig változatlanul hagyja. Ezt az aktivációs függvényt széles körben használjuk, mert képes enyhíteni a gradiensek eltűnésének problémáját.

- **Softmax:**
$$\sigma(x) = \frac{e^{x_k}}{\sum_{i=1}^{n} e^{x_i}},$$
ahol n az osztályok számát, k a k-adik kimeneti neuront jelöli.

Ez az eset gyakran használatos egy neurális hálózat kimeneti rétegében többosztályos osztályozásnál. Ebben az esetben x az utolsó réteg kimenete lehet, erre alkalmazzuk a Softmax-ot. A Softmax 0 és 1 közé skálázza a kimeneti értékeket, majd azok összegét egyre normálja, így tekinthetünk rájuk valószínűségi értékekként is.

Az aktivációs függvény fontos tulajdonsága, hogy segítségével nemlineáris neurális hálózatokat kapunk, ugyanis ezen függvény kihagyásával a neuron számításai egy affin transzformációt alkotnának, amely viszont lineáris lenne. Ekkor viszont hiába építenénk nagy terjedelmű hálózatokat, ez mindig affin transzformációt biztosítana, ezáltal olyan hálózatot kapnánk, amely csupán egy nagy réteg lenne. Hiszen minden réteg esetén számtalan súllyal van szorozva a bemenet, viszont ekkor a bemeneti és kimeneti adatok közti számítást leírhatnánk egyetlen transzformációval is, ha előre elvégezzük az egyes részmátrixok összeszorzását. Azonban, ha minden réteg végén alkalmazunk egy nemlinearitást, az megtöri ezt a folyamatot.

I.3. Definíció. Mesterséges neuronréteg.

Egy mesterséges neuronháló rétege a fentebb definiált mesterséges neuronokból tevődik össze a következőképpen:

Jelölje $f_{\boldsymbol{\theta}_i}$ az egyes neuronokat azonos σ aktivációs függvénnyel, ahol $\boldsymbol{\theta}_i = [\boldsymbol{w}_i, b_i]$ és $i \in (1, \ldots, k)$. Legyen $\boldsymbol{W} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k) \in \mathbb{R}^{n \times k}$ az oszlopvektorokból előállított súlyokból álló mátrix, továbbá $\boldsymbol{b} = (b_1, \ldots, b_k)$ az eltolások vektora.

Ekkor egy k db neuront tartalmazó neuronréteget az alábbi $h_{\boldsymbol{\theta}} : \mathbb{R}^{n \times k} \to \mathbb{R}^k$ függvény definiál:

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = (f_{\boldsymbol{\theta}_1}(\boldsymbol{x}), \ldots, f_{\boldsymbol{\theta}_k}(\boldsymbol{x})) = \sigma(\boldsymbol{W}^T \boldsymbol{x} + \boldsymbol{b}),$$

ahol $\boldsymbol{\theta} = [\boldsymbol{W}, \boldsymbol{b}] = [\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_k]$.

I.4. Definíció. Előrecsatolt neuronháló.

Egy n dimenziós inputtal rendelkező, L rétegű neuronhálót az alábbi függvény definiál:

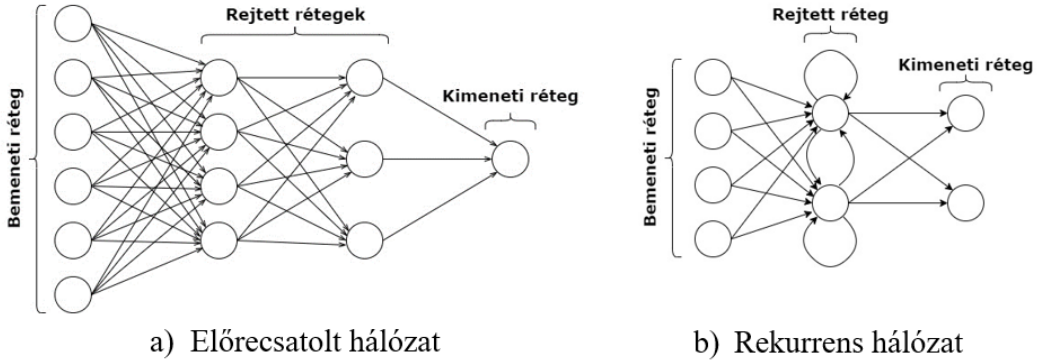
$$\hat{\boldsymbol{y}} = F_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sigma(\boldsymbol{W}_L(\ldots \sigma(\boldsymbol{W}_2(\sigma(\boldsymbol{W}_1 \boldsymbol{x} + \boldsymbol{b}_1) + \boldsymbol{b}_2) \ldots) + \boldsymbol{b}_L).$$

Itt $\hat{\boldsymbol{y}} \in \mathbb{R}^m$, ahol m az utolsó réteg neuronszáma és $\boldsymbol{\theta} = [\boldsymbol{W}_1, \ldots, \boldsymbol{W}_L, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_L]$, σ az aktivációs függvény.
A függvénykompozíciót megadhatjuk egy egyszerűbb alakban is:

$$F_{\boldsymbol{\theta}}(\boldsymbol{x}) = (h_{\boldsymbol{\theta}_L} \circ h_{\boldsymbol{\theta}_{L-1}} \circ \ldots \circ h_{\boldsymbol{\theta}_2} \circ h_{\boldsymbol{\theta}_1})(\boldsymbol{x}),$$

ahol $h_{\boldsymbol{\theta}_l}$ jelöli az l-edik neuronréteget, amelynek paramétereit $\boldsymbol{W}_l$ és $\boldsymbol{b}_l$ tartalmazza.

Egy neurális hálózat tehát számos neuronréteg összekapcsolásából áll elő, melyet matematikai megközelítéssel egy függvénykompozícióként definiálunk.



a) Előrecsatolt hálózat b) Rekurrens hálózat

8. ábra. Neuronok kapcsolásának típusai.

II. A hálózatok rétegei

A biológiai neuronhálók magas szintű bonyolultsága miatt szükséges elvégezni egyfajta leegyszerűsítést, melynek egyik lehetősége a neuronok rétegekbe való rendezése. Rendezéstől függően kétféle neurális hálózatot különböztetünk meg, mindkét esetben valamilyen kapcsolatrendszert definiálva.

Az egyik tipikus architektúra, amikor a kapcsolatok egy irányultsággal vannak ellátva, tehát a bemenettől mindig csak egy irányba, a kimenet felé terjed a számítás. Ezeket a hálózatokat hívjuk előrecsatolt hálózatoknak, melyek különösen hasznosnak bizonyulnak időfüggetlen modellezésnél, például képfelismerésnél.

A másik rendszer dinamikai modellezés esetén használatos. Ekkor szükség van bizonyos visszacsatolásokra, melyek megjegyzik, hogy korábban milyen aktiváció áramlott át a hálózaton. Ezen hálózatokat rekurrens hálózatoknak nevezzük, melyek idősor modellezésénél nyújtanak jobb teljesítményt.

A különbségek mellett egy azonosság is jellemzi a fent említett kétféle neurális hálót, mégpedig az, hogy minden réteg végén egy aktivációs függvényt alkalmazunk. Az előre csatolt hálózatok és a rekurrens hálózatok közötti különbséget szemlélteti a 8. ábra.

III. A neuronháló rétegeinek típusai

A szakdolgozatban csakis előre csatolt hálózatokkal fogunk foglalkozni. Ezen hálózatok esetén két különösen fontos típust különböztetünk meg. A sűrű-, illetve a konvolúciós rétegű hálózatokat.

III.1. Sűrűn kapcsolt rétegek

A mesterséges neurális hálózatok egyik alap építő elemei a sűrűn kapcsolt rétegek. Ezek különösen fontos összetevői az előrecsatolt neurális hálózatoknak. A sűrűn kapcsolt rétegek jellemzője, hogy az előző réteg minden neuronja össze van kötve a következő réteg minden

neuronjával.

III.5. Definíció. Sűrűn kapcsolt réteg.

Ez matematikai megközelítéssel azt jelenti, hogy egy adott $\boldsymbol{x}$ bemenet, $\boldsymbol{W}$ súlyozás és $\boldsymbol{b}$ eltolási vektor esetén egy $\hat{\boldsymbol{y}}$ kimenetű sűrűn kapcsolt réteget az

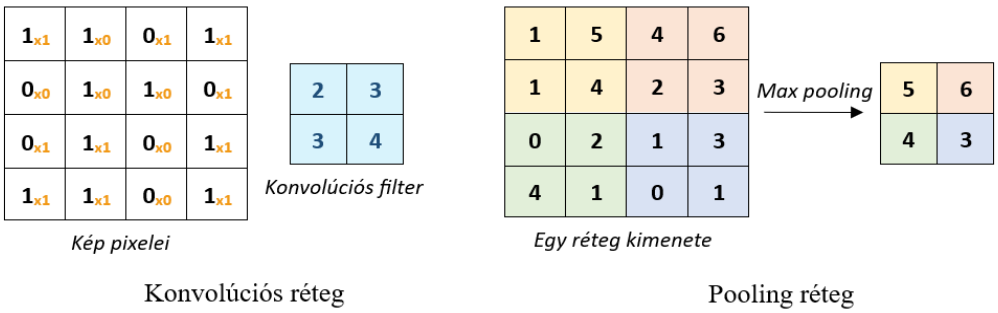
$$\hat{\boldsymbol{y}} = \boldsymbol{x}\boldsymbol{W} + \boldsymbol{b}$$

mátrixszorzás és egy eltolás ad meg.

III.2. Konvolúciós rétegek

A formális leírás előtt induljunk ki az I.4 definíció egyszerűbb alakjából. Ekkor egy előcsatolt L rétegű mesterséges neronháló definíciója paraméterezett függvénykompozícióként: $F_{\boldsymbol{\theta}}(\boldsymbol{x}) = (h_{\boldsymbol{\theta}_L} \circ h_{\boldsymbol{\theta}_{L-1}} \circ \ldots \circ h_{\boldsymbol{\theta}_2} \circ h_{\boldsymbol{\theta}_1})(\boldsymbol{x})$, ahol $h_{\boldsymbol{\theta}_l}$ $(l \in 1, \ldots, L)$ a neuronháló egy konvolúciós rétege. Egy $2D$-s bemeneti kép esetén $h_{\boldsymbol{\theta}_l}$-t konvolúciós rétegnek nevezzük, amennyiben nem minden pixelhez rendel külön súlyt, csupán a pixelek egy halmazához.

A fenti leírás könnyebb megértéséhez vegyük példaként a 9. ábra jobb oldalát:



9. ábra. Konvolúciós réteg és pooling réteg.

Ekkor a bemeneti képet a bal oldal reprezentálja, az egyes pixeleken pirossal jelezve a súlyokat. Erre a képre egy úgynevezett filter kerül — mely esetünkben 3×3-as — és eszerint kerül kiszámításra a konvolúciós réteg. (A második ábra legelső eleme tehát $1 \times 1 + 1 \times 0 + 0 \times 1, ..., 0 \times 1 + 1 \times 1 + 0 \times 0 = 2$ és így tovább.) Egy komplex neuronháló hatékony működéséhez a fent említett filterekből egyszerre többet használunk.

III.6. Definíció. Konvolúciós réteg.

Legyen adott egy $n \times n$ méretű bemenet vagy neuronréteg-kimenet, melyet egy konvolúciós réteg követ. A konvolúciós réteget egy $k \times k$-s filter határoz meg, melyet jelöljön $\boldsymbol{W}$. Ezen esetben a konvolúciós réteg kimeneti mérete $(n - k + 1) \times (n - k + 1)$ lesz.

Ekkor a konvolúció kimenetének egy eleme $\hat{\boldsymbol{y}}_{ij}$ az $\boldsymbol{x}$ bemenete egységeinek súlyozott összege a filteren szereplő súlyozással, a konvolúciós réteg kimenetét pedig az azon alkalmazott aktivációs függvény kimenteként kapjuk, tehát:

$$\hat{\boldsymbol{y}}_{ij} = \sigma\left(\sum_{a=0}^{k-1}\sum_{b=0}^{k-1} \boldsymbol{W}_{ab}\, \boldsymbol{x}_{(i+a)(j+b)}\right).$$

III.3. Pooling rétegek

Egy konvolúciós hálózat felépítéséhez a konvolúciós rétegek mellett szükség van további rétegek beépítésére is. Ezeket az extra rétegeket nevezzük pooling rétegeknek, melyek az $F_{\boldsymbol{\theta}}(\boldsymbol{x})$ felírásban szintén függvénykomponensként szerepelnek. A konvolúciós rétegekhez hasonlóan itt is egy filter halad végig a képeken, ezen esetben dimenzió-csökkentés végett. Pooling rétegnek nevezünk tehát egy olyan réteget, amelynek nincsenek tanítható paraméterei és az előtte szereplő réteg egy részmátrixához egy darab számot rendel. Egy $n \times n$ méretű réteg $k \times k$ méretű filterezésével így egy $\frac{n}{k} \times \frac{n}{k}$ méretű pooling réteget kapunk.

Kétféle fontos pooling réteget különböztetünk meg szerepük szerint. Ezek a max pooling, illetve az average pooling névvel ellátott rétegek, melynél az előbbi maximum-számítást végez, utóbbi pedig átlagolást. Egy max pooling réteg pontos működése látható a fenti 9-es ábra jobb oldalán 2×2-es filterrel, kettes lépésközzel. Azt a paramétert, mely azt határozza meg milyen lépésközökkel mozog a filter, stridenak nevezzük.

III.7. Definíció. Pooling réteg.

A fentebb említett max pooling és average pooling réteg formális definíciója:

$$\boldsymbol{x}_{ij}^{k} = max(\boldsymbol{ip} : (\boldsymbol{ip} + p), \boldsymbol{jq} : (\boldsymbol{jq} + q), k)$$

$$\boldsymbol{y}_{ij}^{k} = avg(\boldsymbol{ip} : (\boldsymbol{ip} + p), \boldsymbol{jq} : (\boldsymbol{jq} + q), k)$$

ahol p jelöli a magasságot, q a szélességet, továbbá $\boldsymbol{x}_{ij}^{k}, \boldsymbol{y}_{ij}^{k}$ a kimeneti értékeket az (i, j, k) pozícióban.

A pooling réteg fontos szerepet játszik a konvolúciós neurális hálózatokban, mivel csökkenti a paraméterek számát és az overfitting kockázatát, miközben megőrzi a fontos, térbeli információkat. Egy 2D-s kép esetén, amely 3 dimenziós tenzorként van reprezentálva (magasság, szélesség, színcsatornák), a pooling réteg a térbeli dimenziókat (magasság és szélesség) csökkenti, miközben megtartja a színcsatornák számát. Ez lehetővé teszi, hogy a hálózat fókuszáljon a fontos jellemzőkre, csökkentve a számítási igényt, és javítva az általánosító képességet.

III.8. Definíció. Normalizációs réteg (BatchNorm).

A kötegelt normalizálás, más néven BatchNorm egy mély tanulásban használt technika a neurális hálózati réteg bemeneteinek normalizálására, melyet bővebben a [Ioffe and Szegedy, 2015] cikk elemez. A bemenetek normalizálásával a BatchNorm segít stabilizálni és felgyorsítani a tanítási folyamatot. A művelet egy adott rétegre:

Legyen adott egy $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ méretű bemeneti mini-batch, ahol minden $\boldsymbol{x}_i$ a függvények egy vektora. Ekkor a mini-batchek átlaga:

$$\boldsymbol{\mu} = \frac{\sum_{i=1}^{N} \boldsymbol{x}_i}{N}$$

Majd ezek varianciája:

$$\boldsymbol{\sigma}^2 = \frac{\sum_{i=1}^{N}(\boldsymbol{x}_i - \boldsymbol{\mu})^2}{N}$$

Ekkor a bemenetek normalizálása:

$$\boldsymbol{x}_i' = \frac{\boldsymbol{x}_i - \boldsymbol{\mu}}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}},$$

ahol ϵ egy egy kis konstans, amelyet a numerikus stabilitás érdekében hozzáadunk a nullával való osztás elkerülése érdekében. A normalizált bemenetek eltolása pedig felírható az

$$\boldsymbol{y}_i' = \upsilon \boldsymbol{x}_i' + \beta$$

képlettel, ahol υ és β tanulható eltolási paraméterek.

A kapott normalizált és transzformált $\boldsymbol{y}_i'$ bemenetek lesznek ezután a BatchNorm réteg kimenetei. Ezen technika fontos tulajdonsága, hogy tanítás során a $\boldsymbol{\mu}, \boldsymbol{\sigma}^2, \upsilon$ és β paraméterek frissítésével a BatchNorm enyhíti a gradiens eltűnésének és felrobbanásának problémáját, melyet a [Santurkar et al., 2018] tanulmány elemez. Ennek jelentése, hogy miközben egy mély hálóban a korábbi rétegek irányában visszafelé terjesztik a hibát és a gradienseket számolják rétegről rétegre, akkor a gradiens könnyen vagy közel 0 vagy hatalmas értékre nőhet. Ez pedig olyan tanulási instabilitáshoz vezethet, amikor loss=nan vagy a legbelső rétegek nem tanulnak.

III.9. Definíció. Reziduális háló.

A reziduális neurális háló, avagy ResNet [He et al., 2016] a mély neurális hálózati architektúra egyik olyan típusa, mely úgynevezett reziduális összekapcsolást vezet be a nagyon mély hálózatok tanításának könnyítése érdekében [He et al., 2016]. Ezt a következő matematikai képlet határozza meg:

$$H_{\boldsymbol{\theta}}(\boldsymbol{x}) = F_{\boldsymbol{\theta}}(\boldsymbol{x}) + \boldsymbol{x},$$

ahol $H_{\boldsymbol{\theta}}(\boldsymbol{x})$ a ResNet kimenetét, $\boldsymbol{x}$ a hálózat bemenetét, $F_{\boldsymbol{\theta}}(\boldsymbol{x})$ pedig a reziduális leképezést, vagyis az $\boldsymbol{x}$-re alkalmazott betanult transzformációt jelöli. A ResNet lényege az $F_{\boldsymbol{\theta}}(\boldsymbol{x})$ reziduális függvény tanulása a mögöttes leképezés közvetlen közelítése helyett. A reziduális leképezés tanulásának megkönnyítése érdekében egy tipikus választás két vagy több konvolúciós rétegből álló blokk, amelyet egy nemlineáris aktivációs függvény, például a ReLU követ. Így az $F_{\boldsymbol{\theta}}(\boldsymbol{x})$ leképezés definiálható

$$F_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{W}_2 \cdot ReLU(\boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}_1) + \boldsymbol{b}_2$$

alakban, ahol $\boldsymbol{W}_1$ és $\boldsymbol{W}_2$ tanulható súlymátrixok, $\boldsymbol{b}_1$, $\boldsymbol{b}_2$ a megfelelő torzítás és ReLU az aktivációs függvény.

III.10. Definíció. Dropout.

A Dropout egy neurális hálózatokban alkalmazott regulációs technika a túlillesztés megelőzésére, mely során a tanítási fázisban véletlenszerűen kiválasztott neuronok kimeneteit

nullázzuk ki adott p valószínűséggel, ahol $0 < p < 1$. Ezáltal a hálózat tanítása során a neurális hálózat különböző és kevésbé függő részhálózatait tanulja meg, csökkentve az egyes neuronokra való túlzott támaszkodást, ami túllillesztéshez vezethet. Ennek a matematikai leírása a következőképpen fogalmazható meg:

$$\boldsymbol{y}_i = (r_i < p) \cdot (\boldsymbol{w}_i \boldsymbol{x}_i + \boldsymbol{b}_i),$$

ahol $\boldsymbol{w}_i$ a súlyokat, x_i a bemeneteket, $\boldsymbol{b}_i$ pedig a torzítást jelöli és r_i egy véletlen szám 0 és 1 között, továbbá $(r_i < p)$ egy bináris érték, ami akkor 1, ha r_i kisebb, mint a dropout valószínűsége p, és 0 különben.

A backpropagation során a gradiens számításánál csak az aktív neuronokat vesszük figyelembe, ezáltal a dropout a tanításnál segít megelőzni a túllillesztést azáltal, hogy csökkenti a neuronok közötti kölcsönös függőséget.

IV. A tanítás folyamata

A neurális hálózatok egyik legfontosabb tulajdonsága az általánosítóképességük. Ez azt jelenti, hogy egy megfelelően nagy és megfelelően konfigurált előrecsatolt neurális hálózat tetszőleges folytonos függvényt tetszőleges pontossággal meg tud közelíteni egy adott hibahatáron belül. Más szóval, egy adott függvényhez, adott hibahatárral, létezik olyan neurális hálózat, amelynek megfelelő paraméterbeállításaival a szóban forgó függvényt meg tudjuk közelíteni.
Ezt a következő tétel mondja ki:

IV.11. Definíció. Univerzális Approximációs Tétel. Adott egy $f : \mathbb{R}^n \to \mathbb{R}^m$ folytonos függvény és egy $\epsilon > 0$ pozitív szám. Ekkor létezik olyan előrecsatolt neurális hálózat, hogy minden $\boldsymbol{x} \in K \subset \mathbb{R}^n$ esetén, ahol K egy kompakt halmaz, az alábbi feltétel teljesül:

$$sup_{\boldsymbol{x} \in K} ||F_{\boldsymbol{\theta}}(\boldsymbol{x}) - f(\boldsymbol{x})|| < \epsilon,$$

ahol $F_{\boldsymbol{\theta}}(\boldsymbol{x})$ jelöli az előrecsatolt neurális hálózat által megközelített függvényt a következő alakban:

$$F_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{i=1}^{N} \boldsymbol{w}_i \cdot \sigma(\boldsymbol{A}_i \boldsymbol{x} + \boldsymbol{b}_i),$$

ahol N a hálózatban lévő neuronok számát, $\boldsymbol{w}_i \in \mathbb{R}^m$ a kimeneti súlyokat, $\boldsymbol{A}_i \in \mathbb{R}^n$ a bemeneti súlyvektorokat, $\boldsymbol{b}_i \in \mathbb{R}$ az eltolásokat, σ pedig egy nemlineáris aktivációs függvényt jelöl.

A tanítási folyamat során a célunk az, hogy megtaláljuk azokat a $\boldsymbol{w}_i, \boldsymbol{A}_i, \boldsymbol{b}_i$ paramétereket, amelyek minimálisra csökkentik a hibát a hálózat kimenete és a kívánt kimenet között. Ezáltal a hálózat a bemeneti adatokhoz alkalmazkodva, a lehető legjobban közelíti a célfüggvényt. A tanítás során használt hibafüggvény vagy veszteségfüggvény azt értékeli, hogy mennyire pontos a hálózat által adott kimenet a kívánt kimenethez képest. A

hibafüggvény minimalizálására többnyire a gradiens alapú optimalizálási algoritmusokat, például a backpropagation módszert használjuk.

Ez a folyamat a hálózat súlyainak fokozatos finomhangolásával történik, amely során a cél az, hogy a hálózat kimenete a lehető legjobban megfeleljen az elvárt kimeneteknek. Az optimális paraméterek megtalálása bonyolult feladat lehet, és gyakran szükség van különböző stratégiákra, például regularizációra vagy hiperparaméterek finomhangolására a túlillesztés elkerülése és a modell általánosítóképességének javítása érdekében.

IV.12. Definíció. Hibafügggvény.

Legyen $\hat{\boldsymbol{y}} = F_{\boldsymbol{\theta}}(\boldsymbol{x})$ a modell által adott kimenet értéke egy adott $\boldsymbol{x}$ bemenetre, ahol $\boldsymbol{x} \in \mathbb{R}^n$ egy bemeneti vektor és $\boldsymbol{\theta}$ jelöli a modell paramétereit. Legyen $\boldsymbol{y}$ az elvárt kimenet értéke ugyanazon $\boldsymbol{x}$ bemenet esetén. Ekkor a veszteségfüggvényt, amely a modell által adott kimenet és az elvárt kimenet közötti eltérést méri, $L(\boldsymbol{y}, F_{\boldsymbol{\theta}}(\boldsymbol{x}))$-ként jelöljük, és a modell paramétereinek optimalizálására törekszünk a következőképpen:

$$\arg\min_{\boldsymbol{\theta}} \sum_{(\boldsymbol{x},\boldsymbol{y})\in D} L(\boldsymbol{y}, F_{\boldsymbol{\theta}}(\boldsymbol{x})),$$

ahol $D = \{(\boldsymbol{x}_i, \boldsymbol{y}_i), i = 1, \ldots, N\}$ az adathalmaz, amely az $\boldsymbol{x}$ bemeneti és $\boldsymbol{y}$ kimeneti értékpárokat tartalmazza.

A háló tanításakor a modell $\boldsymbol{\theta}$ paramétereinek frissítésével törekszünk a hibafüggvény minimalizálására, ezért a folyamatot akár optimalizálási feladatként is definiálhatjuk.

A feladat megoldása során kihasználjuk, hogy a veszteségfüggvény differenciálható a súlyok szerint, ezért a modell tanítása során a gradiens alapú módszereket alkalmazva kiszámítjuk, hogy egyes paraméterek hogyan befolyásolják a veszteséget. Ahogy azt a neuronok működésénél láttuk, a súlyozott összeg, illetve az aktivációs függvény is differenciálható műveletet definiál, így ezek kompozíciója is differenciálható leképezés lesz. A tanítás során minden paraméterhez deriválással kiszámítható, hogy az adott súly milyen mértékben befolyásolja a hibafüggvényt. Ezen derivált értékek egy gradiens vektort alkotnak, amely megmutatja, hogy a jelenlegi paraméterbeállítások függvényében mely irányban növekszik a hiba. A tanulás célja a hiba csökkentése, tehát a paraméterek frissítése a negatív gradiens irányába szükséges. A gyakorlatban a paraméterfrissítést tipikusan egyszerre több adatponton, egy $B \sim D$ mini-batchen hajtjuk végre minden iterációban - a teljes adathalmazon való kiszámítása általában nem is lehetséges nagy adathalmazok esetén. A mini-batchen történő paraméterfrissítés képlete:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \frac{\partial \sum_{(\boldsymbol{x},\boldsymbol{y})\sim B} L(\boldsymbol{y}, F_{\boldsymbol{\theta}}(\boldsymbol{x}))}{\partial \boldsymbol{\theta}},$$

ahol a képletben szereplő η egy előre megválasztott pozitív skalár érték, a tanulási folyamat egy hiperparamétere, melyet tanulási rátának nevezünk és tetszés szerint állítható be. Ez befolyásolja a frissítés lépésméretét. Az optimalizálási folyamat során a tanulási rátát dinamikusan alkalmazkodtathatjuk, kezdetben nagyobb értéket választva a gyorsabb konvergencia érdekében, majd finomítva azt, ahogy a tanulási folyamat halad.

IV.13. Definíció. Tanító adatok.

Legyen $D = \{(\boldsymbol{x}_i, \boldsymbol{y}_i), i = 1, \ldots, N\}$ egy N darab független mintát tartalmazó halmaz, ahol $\boldsymbol{x}_i \in \mathbb{R}^n$ az i. bemeneti vektor, és $\boldsymbol{y}_i$ az ahhoz tartozó elvárt kimenet vagy címke. Minden $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ párt egy közös $P(\boldsymbol{x}, \boldsymbol{y})$ valószínűségi eloszlásból választunk ki, ami az input és output változók együttes eloszlását jelenti.

Felügyelt tanulás esetén a cél az, hogy modellezzük a $P(\boldsymbol{y}|\boldsymbol{x})$ feltételes valószínűségi eloszlást, vagyis készítsünk olyan matematikai modellt, amely képes az adott $\boldsymbol{x}$ bemenethez a megfelelő $\boldsymbol{y}$ kimenetet vagy osztálycímkét hozzárendelni. A tanító adatok ezeket a bemenet-kimenet párokat tartalmazzák, amelyek alapján a gépi tanulási modell tanul és amelyeket a modell a tanulási folyamat során a további, még nem látott adatokon történő predikciókhoz használ.

IV.14. Definíció. Kereszt-entrópia.

A kereszt-entrópia egy gyakran alkalmazott hibafüggvény az osztályozási feladatokban, amely méri az elvárt osztálycímkék és a modell által előrejelzett valószínűségek közötti eltérést. Két fő változata van, attól függően, hogy bináris vagy többosztályos osztályozást végezünk. A bináris osztályozás esetében a kereszt-entrópia hibafüggvénye a következő:

$$L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -(\boldsymbol{y} \log(\hat{\boldsymbol{y}}) + (1 - \boldsymbol{y}) \log(1 - \hat{\boldsymbol{y}})),$$

ahol $\boldsymbol{y}$ jelöli az elvárt kimeneti értéket, mely 0 vagy 1 lehet, $\hat{\boldsymbol{y}}$ pedig az 1-es osztályba tartozás becsült valószínűségét jelöli a modell szerint.

Többosztályos osztályozás esetén a kategorikus kereszt-entrópia hibafüggvényt használjuk:

$$L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\sum_{i=1}^{C} \boldsymbol{y}_i \log(\hat{\boldsymbol{y}}_i),$$

ahol C az osztályok száma, $\boldsymbol{y}$ az elvárt kimeneti értékek vektora (amely egy olyan vektor, aminek csak egy eleme 1, a többi 0, és az 1-es az elvárt osztályt jelöli), és $\hat{\boldsymbol{y}}$ az egyes osztályokba tartozás becsült valószínűségeit tartalmazó vektor a modell szerint.

Mindkét esetben a kereszt-entrópia hibafüggvény segít a modellnek abban, hogy a valós osztálycímkékhez minél közelebbi valószínűségeket jósoljon, így növelve az osztályozás pontosságát. Az optimális modellparaméterek megtalálása érdekében a kereszt-entrópiát minimalizálni kívánjuk a modell tanítása során.

IV.15. Definíció. Klasszifikációs feladat.

A klasszifikációs feladat egy típusa a felügyelt tanulási problémáknak, ahol a cél, hogy az $\boldsymbol{x} \in \mathbb{R}^n$ bemeneti vektorokat egy előre meghatározott kategóriák halmazába, $\{1, 2, \ldots, C\}$-be rendeljük a $F_{\boldsymbol{\theta}}$ paraméterezett függvény segítségével. Itt szintén $\boldsymbol{\theta}$ jelöli a modell paramétereit, és C az osztályok számát. Egy adott bemenet $\boldsymbol{x}_i$ esetében a modell célja, hogy előállítsa az elvárt $\boldsymbol{y}_i$ kimeneti címkét, amely megfelel $\boldsymbol{x}_i$ valódi osztályának. Ez a következőképpen írható le:

$$F_{\boldsymbol{\theta}} : \mathbb{R}^n \to \{1, 2, \ldots, C\},$$

ahol $\hat{y}_i = F_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$ és $\hat{\boldsymbol{y}}_i$ jelöli a modell által prediktált osztálycímkét az i-edik bemeneti vektorhoz. Ekkor $\boldsymbol{x}_i$ célja, hogy minél közelebb legyen a valódi $\boldsymbol{y}_i$ címkéhez.

A $D = \{(\boldsymbol{x}_i, \boldsymbol{y}_i), i = 1, \ldots, N\}$ adathalmazt felhasználva (ahol N az adatpontok száma), a modell tanítása során a $F_{\boldsymbol{\theta}}$ paramétereit úgy optimalizáljuk, hogy minimalizáljuk a predikciók és a valós címkék közötti eltérést, ehhez pedig az egyik leggyakran alkalmazott hibafüggvény a kereszt-entrópia. A modell tanításának célja, hogy az $F_{\boldsymbol{\theta}}$ függvény a tanító adathalmaz alapján megtanulja a bemeneti adatok és a címkék közötti összefüggéseket, majd ezt a tudást alkalmazza új, ismeretlen adatokra is, pontos osztályozást végezve.