**Transfer Learning és lépései**

**Transfer Learning**

A transfer learning (átviteli tanulás) egy gépi tanulási technika, ahol egy előre betanított modellt újrafelhasználunk egy másik, de kapcsolódó feladatra. Ez különösen hasznos akkor, ha az új feladat rendelkezésre álló adatainak mennyisége korlátozott.

**Lépések**

**1. Előre Betanított Modell Kiválasztása:**

* Az első lépés egy olyan modell kiválasztása, amelyet már egy nagy és általános adatbázison (pl. ImageNet) betanítottak.
* Ez a modell már rendelkezik az általános jellemzők kivonásához szükséges képességekkel.

**2. Háló Átalakítása:**

* Az előre betanított modell adaptálása az új feladatra. Ez magában foglalhatja a végső rétegek módosítását vagy hozzáadását, hogy az új feladat specifikus követelményeinek megfeleljen.
* Például, ha a cél egy új kategória klasszifikálása, a végső softmax réteget új osztályokra kell cserélni.

**3. Súlyok Befagyasztása:**

* A súlyok befagyasztása azt jelenti, hogy az előre betanított modell bizonyos rétegeinek súlyait rögzítjük (nem engedjük, hogy ezek a súlyok frissüljenek a tanulás során).
* Általában az alsóbb rétegeket, amelyek az alapvető jellemzőket tanulták meg, fagyasztjuk be, míg a felsőbb rétegeket finomhangoljuk az új adatokkal.

**4. Finomhangolás:**

* A modell finomhangolása az új adatbázison történik. Ez magában foglalja a fagyasztott rétegeken kívüli rétegek újratanítását az új feladat specifikus adataival.
* Fontos a megfelelő tanulási ráta kiválasztása, hogy a finomhangolás során a modell jól illeszkedjen az új adatokra, de ne felejtse el az általános jellemzőket.

**5. Validáció és Tesztelés:**

* A modell teljesítményének értékelése validációs és teszt adatokon.
* Az eredmények alapján további finomhangolás végezhető, ha szükséges.

**Mérlegelendő Szempontok a Háló Átalakításánál és Finomhangolásánál**

**Feladathoz Illeszkedés**

Az új feladat specifikációihoz igazítjuk a modellt. Például egy képfelismerő modell esetében a bemeneti adatok mérete és a kimeneti osztályok száma fontos szempont.

**Adatmennyiség:**

Az új feladathoz rendelkezésre álló adat mennyisége. Ha kevés adat áll rendelkezésre, érdemes több réteget befagyasztani és kevesebb réteget finomhangolni.

**Tanulási Ráta:**

A tanulási ráta megfelelő beállítása elengedhetetlen. Túl nagy tanulási ráta esetén a modell túl gyorsan alkalmazkodik, ami túltanuláshoz vezethet, míg túl alacsony tanulási ráta esetén a tanulás lassú lehet.

**Regularizáció:**

Regularizációs technikák (pl. dropout, L2 regularizáció) használata, hogy elkerüljük a túltanulást az új adatokkal történő finomhangolás során.

**Különbségek Transfer Learningnél Alap Modellhez Képest**

**Előre Tréningezett Modell:**

* **Transfer Learning:** Egy előre tréningezett modellt használunk alapként, amely már egy nagy és általános adathalmazon lett betanítva. Ez a modell már jó alapvető jellemzőket tanult meg.
* **Alapmodell Tanítás:** A nulláról indulva, az összes paraméter kezdeti értéke véletlenszerű vagy nulláról indul, így hosszabb ideig tart, amíg a modell konvergál.

**Finomhangolás (Fine-tuning):**

* **Transfer Learning:** Csak a felsőbb rétegeket vagy az egész modellt finomhangoljuk. Mivel az alsóbb rétegek már jó jellemzőket tanultak meg, általában kevesebb iteráció szükséges.
* **Alapmodell Tanítás:** Az egész modell betanítása szükséges, ami hosszabb időt és több iterációt igényel.

**Early Stopping:**

* **Transfer Learning:** Mivel az alapvető jellemzők már megtanultak, az early stopping gyakran gyorsabban alkalmazható. Az optimalizálási folyamatot korábban leállíthatjuk, amikor a validációs teljesítmény eléri a platót vagy romlani kezd.
* **Alapmodell Tanítás:** Több iteráció szükséges a validációs teljesítmény optimalizálásához, és az early stopping is később történhet meg.

**Hiperparaméterek:**

* **Transfer Learning:** Gyakran kevesebb hiperparaméter finomhangolást igényel, mivel az előre tréningezett modellek általában jól működnek az alapértelmezett hiperparaméterekkel. A finomhangolás során azonban kisebb tanulási rátát szoktak használni.
* **Alapmodell Tanítás:** Több időt és kísérletezést igényel a megfelelő hiperparaméterek beállítása, mivel minden paraméter a semmiből kerül beállításra.

**Mélyhálók és Problémáik: A Gradiens Skálázódása**

**A Gradiens Problémák**

A mély neurális hálózatoknál gyakori probléma a gradiens skálázódása, amely két fő formában jelentkezhet:

**1. Eltűnő Gradiens Probléma (Vanishing Gradient Problem)**

A gradiens eltűnése akkor következik be, amikor a hálózat korai rétegeinek súlyaihoz tartozó gradiens nagyon kicsivé válik. Ennek következtében ezek a rétegek lassan vagy egyáltalán nem tanulnak, mert a gradiens mértéke túl kicsi ahhoz, hogy jelentős súlyfrissítéseket végezzenek.

**Miért történik:**

* A gradiens értéke a hátrafelé terjedés során többszörös szorzásokon megy keresztül. Ha az aktivációs függvények deriváltja kisebb mint 1, akkor a rétegeken keresztül haladva a gradiens értéke exponenciálisan csökkenhet.
* *Például* a sigmoid vagy a tanh aktivációs függvények deriváltjai 0 és 1 közé esnek, így mély hálózatokban a gradiens gyakran nagyon kicsi lesz a korai rétegek számára.

**Következmény:**  
A hálózat tanulási folyamata lelassul vagy megáll, mivel a korai rétegek nem tudnak hatékonyan frissíteni a súlyaikat.

**2. Explodáló Gradiens Probléma (Exploding Gradient Problem)**

A gradiens robbanása akkor következik be, amikor a gradiens értéke a hátrafelé terjedés során nagyon naggyá válik. Ez instabilitást okozhat a hálózat tanulási folyamatában, mivel a súlyfrissítések nagyon nagyok lesznek.

**Miért történik:**

* Ha a súlyok nagy értékeket vesznek fel, vagy ha az aktivációs függvények deriváltjai nagyobbak mint 1, a gradiens értéke exponenciálisan növekedhet a rétegeken keresztül.
* Ennek következtében a gradiens értéke túl nagy lesz, és a súlyfrissítések instabilak lesznek, ami a hálózat paramétereinek szélsőséges változásához vezethet.

**Következmény:**  
A tanulási folyamat során a súlyok instabillá válnak, és a modell teljesítménye romlik vagy teljesen divergens lesz.

**Az Instabil Gradiens Probléma és Elkerülése**

**Batch Normalization (Batch Norm)**

A batch normalization egy olyan technika, amely normalizálja a rétegek bemeneteit egy minibatch-nyi adat alapján. Ez segít stabilizálni és gyorsítani a tanulási folyamatot.

* **Működése:** Az egyes rétegek bemeneteit normalizáljuk úgy, hogy azokat egy meghatározott átlagra és szórásra skálázzuk, majd két további paraméter segítségével újrasúlyozzuk és eltoljuk.

*Másképp: Normalizálja a rétegek bemeneteit, ami stabilizálja és gyorsítja a tanulási folyamatot. A batch normalization segít fenntartani a gradiens értékeit megfelelő tartományban, csökkentve a gradiens eltűnését és robbanását.*

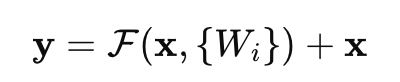
**Reziduális Hálók (Residual Networks, ResNets):**

A reziduális hálók olyan hálózati architektúrák, amelyek reziduális kapcsolatokat (skip connections) használnak. Ezek a kapcsolatok lehetővé teszik, hogy az információ megkerülje az egyik vagy több réteget, közvetlenül egy későbbi rétegbe áramolva.

A reziduális hálók (ResNet) egy speciális neurális hálózati architektúra, amelyet azért fejlesztettek ki, hogy megoldja a mély hálózatokban fellépő gradiens eltűnés (vanishing gradient) és gradiens robbanás (exploding gradient) problémákat. Ezek az architektúrák residual (maradék) kapcsolatokkal vannak felszerelve, amelyek segítenek a gradiens áramlásában a hálózaton keresztül, különösen nagyon mély hálózatokban.

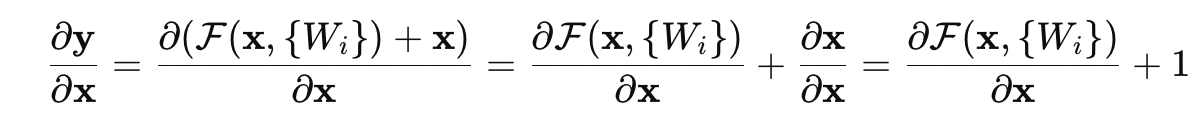
* **Működése:** Ahelyett, hogy a bemenetet közvetlenül a következő rétegbe küldenénk, hozzáadjuk a bemenetet a réteg kimenetéhez, így a kimenet a bemenet és a réteg kimenetének összegét tartalmazza.
* **Előnye:** Segít megőrizni a gradiens áramlását a hálózaton keresztül, csökkentve az eltűnő gradiens problémát és lehetővé téve a nagyon mély hálózatok hatékony tanulását.

*Másképp: A reziduális hálók bevezetése lehetővé teszi, hogy a gradiens könnyebben haladjon a hálózat mélyebb rétegei felé, elkerülve a gradiens eltűnését. A residual kapcsolatok lehetővé teszik, hogy a hálózat tanulása mélyebb rétegekben is hatékony maradjon.*

**Reziduális Kapcsolatok és a Gradiens**  
A ResNet-ben bevezetett residual kapcsolatok segítenek megoldani ezeket a problémákat. Egy residual blokk a következőképpen néz ki:[](https://github.com/Miki0195/4.Felev/blob/main/Neuronh%C3%A1l%C3%B3k/vizsga/residual.png)

Itt *x* a bemenet, F ( x , { W i } ) F(x,{Wi ​ }) pedig a residual függvény, amely tartalmazza a hagyományos konvolúciós rétegeket, aktivációs függvényeket és egyéb műveleteket. A residual blokk kimenete *(y)* a bemenet *(x)* és a residual függvény *(F)* összege.

**Gradiens Visszahaladása Reziduális Hálókban**  
A gradiens visszahaladásának menete:

* **Kimenet Differenciálása:**  
  A kimenetet differenciáljuk a bemenet és a residual függvény szerint:[](https://github.com/Miki0195/4.Felev/blob/main/Neuronh%C3%A1l%C3%B3k/vizsga/gradienshaladas.png)
* **Gradiens Áramlása:**  
  A gradiens áramlása során az összegzés miatt a gradiens nem csökken olyan drasztikusan. Az azonos (1) hozzáadása miatt a gradiens áramlása megőrződik, még akkor is, ha kicsi.
* **Gradiens Mérete:**  
  A gradiens nagyságrendje kevésbé valószínű, hogy eltűnik, mert minden blokkban van egy hozzáadott 1-es érték, ami megőrzi a gradiens nagyságát.

**Miért Stabil a Gradiens Reziduális Hálókban?**

* **Identitás Hozzáadása:**  
  A residual blokkokban a bemenet (x) identitásként van hozzáadva a residual függvény (F) kimenetéhez. Ez az identitás kapcsolat biztosítja, hogy a gradiens áramlása mindig tartalmaz egy közvetlen átviteli útvonalat, amely stabilizálja a gradiens nagyságát.
* **Megakadályozza a Gradiens Eltűnését:**  
  Az x hozzáadása azt jelenti, hogy a gradiens soha nem lehet kisebb, mint 1, így elkerülhető a gradiens eltűnése.
* **Csökkenti a Gradiens Robbanását:**  
  Az identitás kapcsolat segít megőrizni a gradiens nagyságát, és megakadályozza, hogy túl nagy legyen, így elkerülve a gradiens robbanását.