

NYILATKOZAT

Név: Hidy Gábor

ELTE Természettudományi Kar, szak: Matematika BSc.

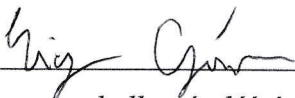
NEPTUN azonosító: BW6B4A

Szakedolgozat címe:

Reziduális neuronhálók és numerikus differenciálegyenlet-közelítések

A **szakedolgozat** szerzőjeként fegyelmi felelősségem tudatában kijelentem, hogy a dolgozatom önálló szellemi alkotásom, abban a hivatkozások és idézések standard szabályait következetesen alkalmaztam, mások által írt részeket a megfelelő idézés nélkül nem használtam fel.

Budapest, 2021. 05. 30.


a hallgató aláírása

Hidy Gábor

Reziduális neuronhálók és numerikus differenciálegyenlet-közelítések

BSc szakdolgozat

Témavezető:

dr. Simon Péter egyetemi tanár

Alkalmazott Analízis és Számításmatematikai Tanszék
Eötvös Loránd Tudományegyetem, Természettudományi Kar
Matematikai Intézet



Eötvös Loránd Tudományegyetem
Természettudományi Kar
2021

Köszönetnyilvánítás

Ez a dolgozat egy félév alatt íródott, de addig a félévig sok évnyi munkán keresztül vezetett az út. Ezért szeretném most megragadni az alkalmat, hogy köszönetet mondjak mindenkinek, aki bármilyen formában is támogatott az életben.

Köszönöm minden tanáromnak és oktatómnak, aki segített az ide vezető úton. Közülük három embernek külön köszönettel tartozom. Ágoston Andrea volt az, aki először megmutatta nekem a matematika szépségeit, a szép, az érdekes és az elgondolkodtató szentháromságát. Sebestyén Ágnes tanárnő továbbvitt ezen az úton, fenntartotta az érdeklődésem és biztosított elegendő kihívással, hogy képességeim is fejlődjenek. És végül, de a legkevésbé sem utolsó sorban dr. Simon Péter, a témavezetőm volt az, aki irányt mutatott és ellátott tanácsokkal, ahogyan a dolgozat vágyból ötletté, majd munkává, és végül valósággá vált.

További köszönet illeti Kovács Jonatán barátomat, akinek sajátos meglátásai sok segítséget nyújtottak a dolgozat csinosításában, a néhol zavaros gondolatok tisztázásában. Köszönöm továbbá édesanyám, Mikes Vivien munkáját, aki hatalmas segítséget nyújtott az ábrák megtervezésében és elkészítésében, kész volt bármikor bármilyen nyelvtani vagy helyesírási kérdésemre válaszolni, és segített az utolsó simítások elvégzésében, kigyomlálta az elgépeléseket, a nyelvhelyességi és helyesírási hibákat.

Köszönöm.

Tartalomjegyzék

Bevezetés	3
1. Differenciálegyenletek közelítése numerikus módszerekkel	4
1.1. Eszköztár és motiváció	4
1.2. Vizsgált numerikus módszerek	6
1.2.1. Explicit Euler-módszer	6
1.2.2. Lineáris többlépéses módszer	8
2. Mesterséges neuronhálók	13
2.1. Általános felépítés	13
2.2. Tanulás	16
2.3. Mély neuronhálók tanításának nehézségei	19
3. Képklasszifikáció mély neuronhálókkal	21
3.1. Képek mint bemeneti adatok	21
3.2. Konvolúciós hálók	23
3.3. Reziduális hálók	25
3.3.1. ResNet architektúrák	26
4. Az LM-ResNet	30
4.1. ResNet mint differenciálegyenlet-közelítés	30
4.2. Implementáció	31
4.3. Eredmények	32
4.4. A kísérletek reprodukálása	33
4.4.1. ResNet eredmények	33
4.4.2. LM-ResNet eredmények	34
Irodalomjegyzék	38

Bevezetés

A mesterséges intelligencia és azon belül is a mesterséges neuronhálók egy egyre nagyobb népszerűségnek örvendő tudományterület a matematika, számítógép-tudomány és a programozás találkozásánál. A mély tanulás (*deep learning*) az elmúlt évtizedben hatalmas löketet adott különböző kutatási területeknek, mint a gépi látás és a természetesnyelv-feldolgozás.

A mesterséges neuronhálók növekedő népszerűségének köszönhetően az elmúlt években megjelent több olyan munka, amely új irányokból próbálja megközelíteni a témát. Az egyik ilyen új megközelítés a mesterséges neuronhálók és a differenciálegyenletek kapcsolatával foglalkozik.

Ennek a dolgozatnak a fő célja az, hogy bemutasson egy konkrét neurális hálót, az LM-ResNetet, amelynek szerkezetét numerikus differenciálegyenlet-megoldási módszerek motiválták. Az LM-ResNet az úgynevezett reziduális hálók egy fajtája, amelyek a képfeldolgozás jelenlegi legerősebb eszközei.

A dolgozat felépítése a következő: az 1. fejezetben bemutatom azokat a numerikus módszereket, amelyek az alapját szolgáltatják a későbbiekben bemutatott hálóknak, illetve közlöm a hozzájuk köthető legfontosabb eredményeket. A 2. fejezetben ismertetem a mesterséges neuronhálók felépítéséhez és tanításához kapcsolódó alapvető fogalmakat.

A 3. fejezetben részletezem a gépi tanulás képklasszifikációhoz használt módszereit, köztük a konvolúciós és azon belül is a reziduális hálókat. Végül a 4. fejezetben bemutatom az LM-ResNet hálót, valamint ismertetem az azt bemutató eredeti cikkben közölt mérések eredményeit. Ezeknek a méréseknek egy részét megismételtem, illetve elvégeztem néhány új mérést, amelyek hiányoztak az eredeti cikkekből. A fejezet végén közlöm az így kapott, saját eredményeimet.

1. fejezet

Differenciálegyenletek közelítése numerikus módszerekkel

Ebben a fejezetben két, differenciálegyenletek közelítésére használt eljárást, az explicit Euler-módszert és a lineáris többlépéses módszert fogom bemutatni, illetve ismertetem néhány alapvető tulajdonságukat. A fejezet nagyrészt két könyvön alapul: az explicit Euler-módszer tárgyalása és a lineáris többlépéses módszer alapvető tulajdonságainak vizsgálata nagyrészt Faragó és Horváth *Numerikus módszerek* [1] című könyvét követi, míg a lineáris többlépéses módszer konvergenciavizsgálatához Hairer, Nørsett és Wanner *Solving Ordinary Differential Equations I: Nonstiff Problems* [2] című könyvét használom fel. Mivel a dolgozat fő fókusza nem e módszerek részletes vizsgálata, sok tételt bizonyítás nélkül közlök; ezek bizonyítása megtalálható a fenti könyvek valamelyikében.

1.1. Eszköztár és motiváció

Az alábbiakban definiálom a differenciálegyenletek általam vizsgált típusát, a kezdetiérték-feladatot, valamint közlök két közismert tételt, amelyek fontosak a differenciálegyenletek, illetve a numerikus közelítések vizsgálatához.

1.1.1. definíció (kezdetiérték-feladat). Legyen $\Omega \subseteq \mathbb{R} \times \mathbb{R}^d$ egy összefüggő nyílt halmaz (tartomány), $(t_0, \mathbf{x}_0) \in \Omega$ pont, és $\mathbf{f} : \Omega \rightarrow \mathbb{R}^d$ folytonos függvény.

Amennyiben $I \subseteq \mathbb{R}$ egy t_0 -t tartalmazó nyílt intervallum, és $\mathbf{x} : I \rightarrow \mathbb{R}^d$ olyan folytonosan differenciálható függvény, amelyre tetszőleges $t \in I$ mellett

$$(t, \mathbf{x}(t)) \in \Omega$$

teljesül, továbbá kielégíti a

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}(t, \mathbf{x}(t)), \quad \mathbf{x}(t_0) = \mathbf{x}_0 \tag{1.1}$$

egyenletet, akkor azt mondjuk, hogy \mathbf{x} a **kezdetiérték-feladat** megoldása.

Megjegyzés. A kezdetiérték-feladatot **Cauchy-feladatnak** is szokás hívni.

Megjegyzés. A továbbiakban az egyváltozós \mathbf{x} függvény deriváltjára az $\dot{\mathbf{x}}$, illetve magasabb rendű deriváltjaira az $\mathbf{x}^{(k)}$ ($k \in \mathbb{N}$) jelölést fogom használni.

Megjegyzés. A fenti definícióban $t_0 = 0$ feltehető. Tekinthejük ugyanis az $\tilde{\mathbf{x}} : t \mapsto \mathbf{x}(t - t_0)$ és $\tilde{\mathbf{f}} : (t, \mathbf{x}) \mapsto \mathbf{f}(t - t_0, \mathbf{x})$ függvényeket, ekkor \mathbf{x} pontosan akkor megoldása a fenti (1.1) egyenletnek, ha $\tilde{\mathbf{x}}$ megoldása az

$$\frac{d\tilde{\mathbf{x}}(t)}{dt} = \tilde{\mathbf{f}}(t, \tilde{\mathbf{x}}(t)), \quad \tilde{\mathbf{x}}(0) = \mathbf{x}_0$$

egyenletnek.

Fontos kérdés a kezdetiérték-problémánál, hogy adott \mathbf{f} , illetve \mathbf{x}_0 mellett létezik-e megoldás, és ez egyértelmű-e. Erre ad választ a következő tétel, speciális tulajdonságú \mathbf{f} esetén.

1.1.2. tétel (Picard–Lindelöf). Legyen $\mathbf{f} : \Omega \rightarrow \mathbb{R}^d$ második változójában lokálisan Lipschitz-tulajdonságú függvény, $(t_0, \mathbf{x}_0) \in \Omega$ tesztölges. Ekkor létezik $r > 0$, hogy az $I = (t_0 - r, t_0 + r)$ intervallumon az (1.1) kezdetiérték-feladatnak egyértelműen létezik megoldása. [3]

A Picard–Lindelöf-tétel bizonyításának alapgondolata az, hogy az (1.1) kezdetiérték-problémával ekvivalens

$$\mathbf{x}(t) = \mathbf{x}_0 + \int_{t_0}^t \mathbf{f}(s, \mathbf{x}(s)) ds$$

integrálegyenlet megoldásának létezését mutatja meg a Banach-fixponttétel segítségével. Mivel a Banach-fixponttétel szokásos bizonyítása egyben egy iterációs algoritmust is megad, lehetséges pusztán a Picard–Lindelöf-tételből közelítőleg megadni a keresett \mathbf{x} függvényt. Ez azonban általában lassú konvergenciát eredményez, és (amennyiben \mathbf{f} primitív függvénye nem ismert) minden iterációban egy integrál közelítését igényli. [4]

Ehelyett ebben a fejezetben más, könnyebben számolható módszerekkel fogok foglalkozni. Ezekhez a módszerekhez érdemes feltenni, hogy \mathbf{f} analitikus, azaz akárhányszor differenciálható. Mivel \mathbf{x} differenciálható és $\dot{\mathbf{x}}(t) = \mathbf{f}(t, \mathbf{x}(t))$, ezért ebből következik, hogy \mathbf{x} is analitikus.

Megjegyzés. \mathbf{f} akárhányszor differenciálható mivoltát többek között az motiválja, hogy mind a differenciálegyenleteknek, mind a mesterséges neuronhálóknak az a fő céljuk, hogy a természetben előforduló, „szép” függvényeket közelítsenek, a „szépségnek” pedig szokásos matematikai megfelelője az analitikusság.

Analitikus függvények alkalmazásának előnye, hogy a numerikus közelítések vizsgálatánál használhatjuk \mathbf{x} Taylor-sorfejtését.

1.1.3. tétel (Lagrange-maradéktagos Taylor-formula). Legyen $I \subseteq \mathbb{R}$ nyílt intervallum, $\tau \in I$ és $x : I \rightarrow \mathbb{R}$ akárhányszor differenciálható függvény. Ekkor tetszőleges $t \in I$ és $n \in \mathbb{N}$ esetén létezik $\xi \in [\tau, t]$, hogy

$$x(t) = \sum_{k=0}^n \frac{x^{(k)}(\tau)}{k!} (t - \tau)^k + \frac{x^{(n+1)}(\xi)}{(n+1)!} (t - \tau)^{n+1} \quad (1.2)$$

1.1.4. következmény. Az (1.1.3) tétel feltételei mellett tetszőleges $[a, b] \subset I$ kompakt intervallum, $t \in [a, b]$ és $n \in \mathbb{N}$ esetén

$$x(t) = \sum_{k=0}^n \frac{x^{(k)}(\tau)}{k!} (t - \tau)^k + \mathcal{O}((t - \tau)^{n+1}) \quad (1.3)$$

Most térjünk vissza a vektorváltozós \mathbf{x} esetre. Ekkor az 1.1.3 Taylor-formula és annak 1.1.4 következménye koordinátánként teljesülnek, így kimondható az alábbi tétel:

1.1.5. tétel (Taylor-formula vektorértékű függvényekre). Ha $I \subseteq \mathbb{R}$ nyílt intervallum, $\tau \in I$, és $\mathbf{x} : I \rightarrow \mathbb{R}^d$ akárhányszor differenciálható függvény, akkor tetszőleges $[a, b] \subset I$ kompakt intervallum, $t \in [a, b]$ és $n \in \mathbb{N}$ esetén

$$\mathbf{x}(t) = \sum_{k=0}^n \frac{\mathbf{x}^{(k)}(\tau)}{k!} (t - \tau)^k + \mathcal{O}((t - \tau)^{n+1}) \quad (1.4)$$

Megjegyzés. Az \mathcal{O} jelölést általában egyváltozós függvények esetén szokás alkalmazni. Én itt egy kicsit általánosabb értelemben használom: a g és h vektorértékű függvényekre $g(x) = \mathcal{O}(h(x))$, ha $\|g(x)\| = \mathcal{O}(\|h(x)\|)$ a hagyományos értelemben.

1.2. Vizsgált numerikus módszerek

Ebben a részben bemutatok két numerikus módszert a kezdetiérték-probléma megoldására, és megvizsgálom az alapvető tulajdonságait.

Mindig az (1.1) egyenlettel definiált kezdetiérték-feladat megoldását szeretném közelíteni $t_0 = 0$ mellett egy $[0, T]$ zárt intervallumon, amely része egy olyan I nyílt intervallumnak, ahol a megoldás egyértelmű. Lerögzíték egy $n \in \mathbb{N}$ számot, és veszem az

$$\omega_n = \left\{ i\Delta t : 0 \leq i \leq n, \Delta t = \frac{T}{n} \right\} \quad (1.5)$$

rácshálót. Legyen $t_i = i\Delta t$.

Ezekben a t_i pontokban akarom numerikusan közelíteni \mathbf{x} -et, valamilyen $\mathbf{x}_i \approx \mathbf{x}(t_i)$ értékekkel. ($0 \leq i \leq n$.) Ezek közül $\mathbf{x}_0 = \mathbf{x}(t_0)$ választható, hiszen ennek értéke ismert az (1.1) egyenlet alapján. A tárgyalt módszerek iteratív módon fogják meghatározni az \mathbf{x}_i értékeket, az \mathbf{x}_0 értékből kiindulva.

1.2.1. Explicit Euler-módszer

Az első vizsgált módszer mögött a következő az intuíció: tegyük fel, hogy $\mathbf{x}_i \approx \mathbf{x}(t_i)$ egy elég jó közelítés. Ekkor megfelelően kicsi Δt_i mellett

$$\frac{\mathbf{x}(t_i + \Delta t) - \mathbf{x}_i}{\Delta t} \approx \dot{\mathbf{x}}(t_i)$$

teljesül. Mivel $\mathbf{x}(t_i + \Delta t) = \mathbf{x}(t_{i+1})$, és $\dot{\mathbf{x}}(t_i) = \mathbf{f}(t_i, \mathbf{x}(t_i))$, ezért $\mathbf{x}(t_{i+1}) \approx \mathbf{x}_i + \mathbf{f}(t_i, \mathbf{x}(t_i))$. Ráadásul mivel \mathbf{f} Lipschitz-tulajdonságú a második változójában, ezért $\mathbf{f}(t_i, \mathbf{x}(t_i)) - \mathbf{f}(t_i, \mathbf{x}_i) = \mathcal{O}(\mathbf{x}(t_i) - \mathbf{x}_i)$. Így ezzel azt kaptuk, hogy

$$\mathbf{x}(t_{i+1}) \approx \mathbf{x}_i + \Delta t \mathbf{f}(t_i, \mathbf{x}_i)$$

Mivel a cél, hogy $\mathbf{x}_{i+1} \approx \mathbf{x}(t_{i+1})$ teljesüljön, ezért egy – heurisztikusan – jó becslést kapunk, ha \mathbf{x}_{i+1} -et a fenti becslés jobb oldalának választjuk meg. Az explicit Euler-módszer pont ezt a becslést használja, amely, mint ki fog derülni, nemcsak heurisztikusan, hanem precíz értelemben is „jó”, azaz a közelítőértékek a pontos megoldáshoz fognak konvergálni.

1.2.1. definíció (explicit Euler-módszer). Azt a módszert vizsgáljuk tehát, ahol tetszőleges $0 \leq i \leq n - 1$ mellett

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \Delta t \mathbf{f}(t_i, \mathbf{x}_i) \quad (1.6)$$

Megjegyzés. A módszer **explicit**, mert \mathbf{x}_{i+1} közvetlenül számolható \mathbf{x}_i ismeretében.

A konvergenciavizsgálathoz szükségem van arra, hogy tudjam mérni a hibát. Így bevezetem az

$$\mathbf{e}_i = \mathbf{x}_i - \mathbf{x}(t_i) \quad (1.7)$$

hibatagot. (Ekkor $\mathbf{e}_0 = \mathbf{0}$.) Ez a hiba alapvetően két forrásból fog származni. Egyrészt az explicit Euler-módszer $\Delta t \mathbf{f}(t_i, \mathbf{x}_i) = \mathbf{x}_{i+1} - \mathbf{x}_i$ összefüggése nem marad meg, ha az \mathbf{x}_i és \mathbf{x}_{i+1} becsléseket helyettesítjük a pontos $\mathbf{x}(t_i)$ és $\mathbf{x}(t_{i+1})$ értékekkel. Másrészt \mathbf{x}_{i+1} kiszámolásánál $\dot{\mathbf{x}}(t_i) = \mathbf{f}(t_i, \mathbf{x}(t_i))$ helyett az $\mathbf{f}(t_i, \mathbf{x}_i)$ közelítést használjuk. Ezeknek mérésére bevezetem a

$$\mathbf{g}_i = \Delta t \mathbf{f}(t_i, \mathbf{x}(t_i)) - (\mathbf{x}(t_{i+1}) - \mathbf{x}(t_i)), \quad \mathbf{q}_i = \mathbf{f}(t_i, \mathbf{x}_i) - \mathbf{f}(t_i, \mathbf{x}(t_i)) \quad (1.8)$$

mennyiségeket.

Ekkor a különböző becslések között felírható az alábbi összefüggés:

1.2.2. állítás (hibaegyenlet).

$$\mathbf{e}_{i+1} - \mathbf{e}_i = \mathbf{g}_i + \Delta t \mathbf{q}_i \quad (1.9)$$

Bizonyítás.

$$\mathbf{e}_{i+1} - \mathbf{e}_i = (\mathbf{x}_{i+1} - \mathbf{x}(t_{i+1})) - (\mathbf{x}_i - \mathbf{x}(t_i)) = -(\mathbf{x}(t_{i+1}) - \mathbf{x}(t_i)) + \Delta t \mathbf{f}(t_i, \mathbf{x}_i)$$

Ekkor ha az első taghoz hozzáadunk $\Delta t \mathbf{f}(t_i, \mathbf{x}(t_i))$ -t, a második tagból pedig levonunk, akkor pont a kívánt állítást kapjuk. \square

1.2.3. tétel. Ha \mathbf{f} második változójában Lipschitz-tulajdonságú $L > 0$ konstanssal, akkor létezik $M \in \mathbb{R}$, hogy

$$\|\mathbf{e}_n\| \leq \Delta t \cdot M T e^{LT} \quad (1.10)$$

Bizonyítás. Az \mathbf{f} függvény Lipschitz-tulajdonsága miatt $\|\mathbf{q}_i\| \leq L \|\mathbf{e}_i\|$. Ekkor az (1.9) hibaegyenletre felírt háromszög-egyenlőtlenségből indukcióval látszik, hogy

$$\|\mathbf{e}_n\| \leq \sum_{i=0}^{n-1} (1 + \Delta t L)^i \|g_{n-1-i}\| < (1 + \Delta t L)^n \sum_{i=0}^{n-1} \|\mathbf{g}_{n-1-i}\| < e^{TL} \sum_{i=0}^n \|\mathbf{g}_{n-1-i}\|$$

Az (1.4) Taylor-formula másodrendű alakja miatt létezik $M \in \mathbb{R}$, hogy $\|\mathbf{g}_i\| = M \Delta t^2$, így

$$\|\mathbf{e}_n\| < e^{TL} \sum_{i=0}^{n-1} \|\mathbf{g}_{n-1-i}\| \leq e^{TL} T M \Delta t$$

\square

Ebből a tételből könnyen láthatóan következik az Euler-módszer konvergenciája a T pontban, illetve a $[0, T]$ intervallum tetszőleges pontjában.

1.2.4. következmény. A T pontban az (1.6) explicit Euler-módszer $n \rightarrow \infty$ mellett elsőrendben konvergens az ω_n ekvidisztáns rácshálók sorozatán, azaz $\mathbf{x}_n - \mathbf{x}(T) = \mathcal{O}(\Delta t)$.

1.2.5. következmény. $n \in \mathbb{N}$ esetén legyen \mathbf{y}_n az \mathbf{x} függvény ω_n rácshálón vett (1.6) explicit Euler-módszer által adott közelítéseinek lineáris interpolációja. Ekkor tetszőleges $t \in [0, T]$ esetén $\mathbf{y}_n(t) \rightarrow \mathbf{x}(t)$ elsőrendben.

1.2.2. Lineáris többlépéses módszer

Az explicit Euler-módszer **egylépéses módszer** volt, mert \mathbf{x}_{i+1} meghatározásához csak az \mathbf{x}_i értéket használta fel. A következő módszer bemutatásához rögzíték egy $m \in \mathbb{N}$ számot. A következőkben egy olyan módszercsaládot definiálok, amelyben \mathbf{x} egy adott pontbeli értékét az előző m közelítőérték segítségével approximálom. A módszercsalád tagjai tehát **m -lépéses módszerek** lesznek.

1.2.6. definíció (lineáris többlépéses módszer). Legyenek $\alpha_0, \alpha_1 \dots \alpha_m$, illetve $\beta_0, \beta_1 \dots \beta_m$ rögzített együtthatók. Ekkor a lineáris m -lépéses módszer i . közelítő értéke ($i \geq m$) $\mathbf{x}_{i-m}, \mathbf{x}_{i-m+1}, \dots, \mathbf{x}_{i-1}$ ismeretében az

$$\sum_{k=0}^m \alpha_k \mathbf{x}_{i-k} = \Delta t \left(\sum_{k=0}^m \beta_k \mathbf{f}(t_{i-k}, \mathbf{x}_{i-k}) \right) \quad (1.11)$$

egyenlet megoldása.

Ha a β_0 paraméter 0, a módszer **explicit**, ha $\beta_0 \neq 0$, a módszer **implicit**.

Megjegyzés. A fenti módszer \mathbf{x}_i -t az $\mathbf{x}_{i-m}, \mathbf{x}_{i-m+1}, \dots, \mathbf{x}_{i-1}$ értékek ismeretében határozza meg, ezért valójában a módszer alkalmazásához szükséges megadni \mathbf{x}_0 mellett az $\mathbf{x}_1, \mathbf{x}_2 \dots \mathbf{x}_{m-1}$ közelítéseket. Ezeket a gyakorlatban általában valamilyen más numerikus módszer segítségével határozzák meg.

Érdeemes megjegyezni, hogy bár az (1.11) képletben egyenlőség szerepel, implicit esetben ha \mathbf{f} nem speciális alakú, akkor \mathbf{x}_{i+1} értékét is csak valamilyen közelítő algoritmussal, például Newton-módszerrel lehetséges meghatározni.

A továbbiakban felteszem, hogy $\alpha_0 \neq 0$. Nyilvánvaló, hogy ha az α_k és β_k paraméterek mindegyikét beszorozzuk valamilyen nemnulla skalárral, az nem változtat az \mathbf{x}_i közelítő-értékeken. Így a módszer nem veszít az általánosságából, ha a továbbiakban az

$$\alpha_0 = 1$$

feltevéssel élek.

Megjegyzés. Ahogyan azt a nevezéktan is sugallja, az egylépéses módszerek az m -lépéses módszer $m = 1$ esetei. Az explicit Euler-módszer megfogalmazható például az (1.11) alakban, $\alpha_1 = -1, \beta_0 = 0, \beta_1 = 1$ választással.

Hibabecslés és konzisztencia

Vizsgáljuk most meg a lineáris m -lépéses módszer hibáját. A hiba fő forrása minden esetben az lesz, hogy a keresett \mathbf{x} függvény értéke a t_i helyen csak közelítőleg egyezik meg \mathbf{x}_i -vel, ezért az (1.11) képletben \mathbf{x}_{i-k} helyére $\mathbf{x}(t_{i-k})$ -t írva az egyenlőség is csak közelítőleg fog teljesülni. Ennek a mérésére bevezeték egy új mennyiséget.

1.2.7. definíció (lokális approximációs hiba). A $\Delta t, \alpha_1, \alpha_2 \dots \alpha_m$, valamint $\beta_0, \beta_1 \dots \beta_m$ paraméterekkel definiált lineáris m -lépéses módszer t_i -beli lokális approximációs hibája

$$\mathbf{g}_i(\Delta t) = \sum_{k=0}^m \left(\alpha_k \mathbf{x}(t_{i-k}) - \Delta t \beta_k \mathbf{f}(t_{i-k}, \mathbf{x}(t_{i-k})) \right) \quad (1.12)$$

1.2.8. definíció. A lineáris többlépéses módszer p -edrendű, ha $\mathbf{g}_i(\Delta t) = \mathcal{O}(\Delta t^{p+1})$.

1.2.9. definíció. Konzisztensnek nevezek egy lineáris többlépéses módszert, ha legalább elsőrendű, azaz ha $\mathbf{g}_i(\Delta t) = \mathcal{O}(\Delta t)$.

Az alábbi tétel azt mutatja, hogy a p -edrendűséghez elég arról gondoskodni, hogy az $\alpha_1, \alpha_2 \dots \alpha_m$ és $\beta_0, \beta_1 \dots \beta_m$ paraméterekre teljesüljenek bizonyos összefüggések.

1.2.10. tétel. Az (1.11) lineáris többlépéses módszer pontosan akkor p -edrendű, ha a paraméterek teljesítik a

$$\sum_{k=0}^m \alpha_k = 0, \quad \frac{1}{\ell} \sum_{k=0}^m k^\ell \alpha_k + \sum_{k=0}^m k^{\ell-1} \beta_k = 0 \quad (1.13)$$

feltételeket tetszőleges $1 \leq \ell \leq p$ mellett.

A tétel bizonyításához a t_i körüli (1.4) Taylor-formula felírása szükséges p -edrendben $\mathbf{x}(t_{i-k})$ -ra, illetve $p-1$. rendben $\dot{\mathbf{x}}(t_{i-k})$ -ra. Innen a bizonyítás elemi számolással adódik.

Kérdés természetesen, hogy adott m mellett ezek a feltételek milyen p -re teljesíthetők. Összesen $2m+1$ paraméterre akarok $p+1$ feltételt kikötni, aminek szükséges feltétele, hogy $p \leq 2m$ teljesüljön, illetve explicit egyenlet esetén $p \leq 2m-1$. A következőkben [4] alapján egy alsó és felső korlát közé szorítom p maximális értékét.

Az alábbi következmény bizonyítása szintén elemi, a β_k értékekre felírt lineáris egyenletrendszer megoldhatóságán alapszik.

1.2.11. következmény. Ha adott $\alpha_1, \alpha_2, \dots, \alpha_m$ számok teljesítik a

$$\sum_{k=0}^m \alpha_k = 0$$

feltételt, akkor ezekhez egyértelműen választhatók olyan $\beta_0, \beta_1, \dots, \beta_m$ számok, hogy az általuk meghatározott lineáris többlépéses módszer $m+1$ -edrendű.

Világos, hogy explicit módszer esetén az (1.13) feltételek mellé még a $\beta_0 = 0$ feltételt is fel kell venni, így ekkor a fenti módszerrel csak m -edrendűség mutatható meg. Germund Dahlquist 1956-ban belátta, hogy ez explicit módszer esetén éles, implicit módszer esetén pedig legfeljebb 1-gyel javítható.

1.2.12. tétel (Dahlquist első tétele). Az (1.11) lineáris m -lépéses módszer p rendjére teljesül

1. $p \leq m+2$,
2. $p \leq m+1$, ha p páratlan, és
3. $p \leq m$, ha a módszer explicit.

Megjegyzés. Ahhoz, hogy egy m -lépéses módszer valóban jó közelítést adjon, szükséges, hogy az első m értéket a pontos megoldáshoz elég közel adjuk meg. A gyakorlatban gyakran $k < m$ esetén az \mathbf{x}_k közelítés megadására egy megfelelő rendű $k-1$ lépéses módszert alkalmaznak.

Szükséges feltétel konvergenciára

A többlépéses lineáris módszer konvergenciáját két speciális polinomon keresztül fogom vizsgálni.

1.2.13. definíció (stabilitási polinomok). Az (1.11) egyenlettel leírt lineáris m -lépéses módszer első stabilitási polinomja ϱ , második stabilitási polinomja σ , ahol

$$\varrho(\zeta) = \sum_{k=0}^m \alpha_k \zeta^{m-k}, \quad \sigma(\zeta) = \sum_{k=0}^m \beta_k \zeta^{m-k} \quad (1.14)$$

Ekkor az (1.13) feltétel $p = 1$ esete, azaz a módszer konzisztenciája ekvivalens a

$$\varrho(1) = 0, \quad \dot{\varrho}(1) = \sigma(1) \quad (1.15)$$

feltétellel.

1.2.14. definíció (gyökfeltétel). Egy polinom kielégíti a gyökfeltételt, ha minden gyöke a zárt egységkörbe esik, és a peremen minden gyöke egyszeres.

1.2.15. definíció (0-stabilitás). Az (1.11) lineáris többlépéses módszer 0-stabil, ha az első stabilitási polinomja kielégíti a gyökkritériumot.

A 0-stabilitás fontos eszköze lesz a többlépéses módszer pontosságának vizsgálatához. Meg fogom ugyanis mutatni, ha egy p -edrendű módszer 0-stabil, akkor kis Δt mellett elég jól közelíti az \mathbf{x} pontos megoldást. Az alábbi konvergenciafogalmak mutatják meg, hogy mit jelent az „elég jó” közelítés.

1.2.16. definíció (konvergencia). Az (1.11) lineáris többlépéses módszer konvergens egy T pontban az (1.5) szerint definiált ω_n rácshálósorozaton, ha tetszőleges olyan $\mathbf{x}_{1,n}, \mathbf{x}_{2,n}, \dots, \mathbf{x}_{m-1,n}$ kezdőérték-sorozat mellett, amelyekre $1 \leq k \leq m-1$ esetén $\mathbf{x}_{k,n} \rightarrow \mathbf{x}_0$ teljesül, a T pont belüli közelítés tart \mathbf{x} T -ben felvett értékéhez, azaz $\mathbf{x}_n \rightarrow \mathbf{x}(T)$.

A módszer p -edrendben konvergens, ha létezik olyan $\Delta t_0, C_1, C_2 > 0$, hogy tetszőleges $\Delta t \leq \Delta t_0$ és $\mathbf{x}_{1,n}, \mathbf{x}_{2,n}, \dots, \mathbf{x}_{m-1,n}$ kezdőértékekre, ha minden $1 \leq k \leq m-1$ mellett teljesül az $\|\mathbf{x}(k\Delta t) - \mathbf{x}_{k,n}\| \leq C_1 \Delta t^p$ feltétel, akkor $\|\mathbf{x}(T) - \mathbf{x}_n\| \leq C_2 \Delta t^p$ is teljesül.

A konzisztencia és a 0-stabilitás konvergenciájával való kapcsolatáról biztosít minket először a következő tétel.

1.2.17. tétel. Ha az (1.11) lineáris többlépéses módszer tetszőleges T pontban konvergens az ω_n rácshálók sorozatán, akkor a módszer 0-stabil és konzisztens.

Felírás mint egylépéses módszer

A következőkben vázolni fogom a másik irányú implikációt, azaz hogy a konzisztenciából és 0-stabilitásból következik a konvergencia. Ehhez érdemes átírni az (1.11) lineáris m -lépéses módszert egy magasabb dimenzióbeli egylépéses módszerré. Ehhez legyen $X_i \in \mathbb{R}^{dm}$ az $X_i = (\mathbf{x}_{i-1}, \mathbf{x}_{i-2}, \dots, \mathbf{x}_{i-m})^\top$ vektor, és legyen $\boldsymbol{\psi}_i(t_i, U_i, \Delta t) \in \mathbb{R}^d$ a

$$\boldsymbol{\psi}_i(t_i, X_i, \Delta t) = \beta_0 \mathbf{f} \left(t_i, \Delta t \boldsymbol{\psi}_i - \sum_{k=1}^m \alpha_k \mathbf{x}_{i-k} \right) + \sum_{k=1}^m \beta_k \mathbf{f}(t_{i-k}, \mathbf{x}_{i-k}) \quad (1.16)$$

implicit egyenlet által definiált függvény,

$$A = \begin{bmatrix} -\alpha_1 & -\alpha_2 & \cdots & -\alpha_{m-1} & -\alpha_m \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & 0 \\ 0 & \cdots & 0 & 1 & 0 \end{bmatrix} \quad (1.17)$$

$m \times m$ -es mátrix.

Az A mátrix mellett az egy lépéses módszer felírásához szükség lesz mátrixok Kronecker-szorzatának definiálására, illetve annak egy egyszerű tulajdonságára.

1.2.18. definíció (Kronecker-szorzat). Legyen P $m_1 \times d_1$ -es, Q $m_2 \times d_2$ -es mátrix. Ekkor P és Q Kronecker-szorzatának a

$$\begin{bmatrix} p_{11}Q & p_{12}Q & \cdots & p_{1d_1}Q \\ p_{21}Q & p_{22}Q & \cdots & p_{2d_1}Q \\ \vdots & \vdots & \ddots & \vdots \\ p_{m_1 1}Q & p_{m_1 2}Q & \cdots & p_{m_1 d_1}Q \end{bmatrix}$$

$m_1 m_2 \times d_1 d_2$ -es blokkmátrixot nevezünk. (Jele: $P \otimes Q$). [5]

A Kronecker-szorzat segítségével és a $\Phi_i(t_i, X_i, \Delta t) = (e_1 \otimes I)\psi_i(t_i, X_i, \Delta t)$ jelölés bevezetésével – ahol e_1 az \mathbb{R}^m -beli első koordinátavektor, I pedig a d dimenziós identitásmátrix – az (1.11) lineáris többlépéses módszer felírható

$$X_{i+1} = (A \otimes I)X_i + \Delta t \Phi_i(t_i, X_i, \Delta t) \quad (1.18)$$

alakban. Legyen G az $X_i \mapsto (A \otimes I)X_i + \Delta t \Phi_i(t_i, X_i, \Delta t)$ függvény, ekkor az (1.18) módszer $X_{i+1} = G(X_i)$ alakban írható fel. Ez a felírás egy egy lépéses közelítése az

$$X(t) = (\mathbf{x}(t - \Delta t), \mathbf{x}(t - 2\Delta t), \dots, \mathbf{x}(t - m\Delta t))^T \quad (1.19)$$

függvénynek.

A következő két lemma az (1.11) lineáris többlépéses módszer rendjének és az (1.18) származtatott egy lépéses módszer hibájának kapcsolatát írja le. Az első lemma állítása könnyű számolással ellenőrizhető. A második lemma bizonyításának fő gondolata az, hogy A sajátértékei és ϱ gyökei megegyeznek, ezek után pedig néhány, lineáris algebrából és numerikus analízisből jól ismert összefüggést kell alkalmazni.

1.2.19. lemma. Ha az (1.11) lineáris többlépéses módszer p -edrendű, $m \leq i \leq n$. Ekkor létezik $M \in \mathbb{R}$ i -től független konstans, amelyre teljesül, hogy $\|X(t_{i+1}) - G(X(t_i))\| \leq M\Delta t^{p+1}$.

Megjegyzés. Fontos, hogy a fenti lemma állítása normafüggetlen, hiszen \mathbb{R}^{dm} -ben bármely két norma ekvivalens.

1.2.20. lemma. Ha az (1.11) lineáris többlépéses módszer 0-stabil, akkor létezik $\|\cdot\|_*$ vektornorma \mathbb{R}^{dm} -en, hogy a belőle származtatott mátrixnorma szerint $\|A \otimes I\|_* \leq 1$ teljesül.

A két lemma segítségével már ki tudjuk mondani a lineáris többlépéses módszer konvergenciájáról szóló fő tételt.

1.2.21. tétel. Ha az (1.11) lineáris többlépéses módszer p -edrendű és 0-stabil, akkor p -edrendben konvergencia tetszőleges T -ben az (1.5) képlettel felírt ω_n rácshálók mentén.

Bizonyítás. Legyen L az \mathbf{f} második változójához tartozó Lipschitz-konstans az $\|\cdot\|_*$ normában. \mathbf{f} Lipschitz-tulajdonsága miatt L σ -nek is Lipschitz-konstansa a második változóban, így az 1.2.20. lemma miatt

$$\|G(X_i) - G(Y_i)\|_* \leq \|A \otimes I\|_* \|X_i - Y_i\|_* + \Delta t L \|X_i - Y_i\|_* \leq (1 + \Delta t L) \|X_i - Y_i\|_*$$

Az 1.2.19. lemma szerint létezik M , hogy $\|X(t_{i+1}) - G(X(t_i))\|_* \leq M \Delta t^{p+1}$. Innen indukcióval és egyszerű becslésekkel belátható, hogy

$$\|X(t_n) - X_n\|_* \leq \|X(t_m) - X_m\|_* e^{TL} + \frac{M}{L} \Delta t^p (e^{TL} - 1)$$

□

Mint azt a 4. fejezetben látni fogjuk, a differenciálegyenletek Euler-diszkretizációja és a reziduális neuronhálók között van egy természetes összefüggés. Így az $\dot{\mathbf{x}}(t) = \mathbf{f}(t, \mathbf{x}(t))$, illetve speciálisan az $\dot{\mathbf{x}}(t) = \mathbf{f}(\Theta(t), \mathbf{x}(t))$ alakú differenciálegyenletek egyfajta végtelen mély reziduális hálót modelleznek, így a véges mélységű hálók egy-egy ilyen egyenlet megoldásának approximációi.

Ebben a fejezetben láttuk, hogy a fenti alakú differenciálegyenletek megoldásának nem az explicit Euler-módszer az egyetlen közelítése, többek között a lineáris többlépéses módszerek gyakran jobb közelítést is eredményeznek, hiszen a megfelelő feltételek mellett magasabb rendű konvergencia is igaz. A 4. fejezetben bemutatott LM-ResNet architektúra fő gondolata az lesz, hogy egy háléhoz tartozó differenciálegyenletet az Euler-módszer helyett egy lineáris kétlépéses módszerrel közelíti, amely az itt tárgyaltak szerint legalább olyan jó közelítést kell hogy adjon, mint az Euler-módszert használó ResNet modellek. Mint az majd kiderül, valójában a mérések arra utalnak, hogy nemhogy nem gyengébb, de egyenesen erősebb modellt eredményez az új közelítő módszer alkalmazása.

2. fejezet

Mesterséges neuronhálók

Ennek a fejezetnek a célja, hogy bemutassa a mesterséges neuronhálók és azon belül is a mély hálók vizsgálatához tartozó alapvető fogalmakat és eszköztárat. A mesterséges neuronhálók vizsgálata egy sokkal inkább gyakorlati, mint elméleti terület, ezért ez a dolgozat is inkább gyakorlati megközelítést fog használni. Ennek ellenére természetesen vannak elméleti eredmények, ezek egy részét bizonyítás nélkül közölni fogom, megjelölve a forrásokat, ahol megtalálható a kimondott állítások részletes bizonyítása.

Ennek, valamint a következő fejezetnek a nagy része Aggarwal *Neural Networks and Deep Learning* [6] című könyvén alapul. Az első részhez ezenkívül gyakran használom Álmos, Győri, Horváth és Kóczy *Genetikus algoritmusok* [7] könyvének a mesterséges intelligenciával foglalkozó fejezetét is, amely szépen összefoglalja a témához tartozó alapvető definíciókat, valamint átülteti magyarrá a szakkifejezések egy részét.

2.1. Általános felépítés

Egy mesterséges neuronháló egy speciális alakú függvény, amelyet legtöbbször más függvények közelítésére használnak. Ennyiben tehát nem különbözik egy analitikus függvény Taylor-polinomjaitól vagy egy folytonos függvény Lagrange-interpolációjától.

A fő különbség abból fog adódni, hogy míg a fenti közelítő módszerek elemi függvények összegeivel és szorzataival közelítették a célfüggvényt, a mesterséges neuronhálók ugyanezt függvénykompozícióval teszik. Ezek a függvények a háló úgynevezett rétegei, a rétegek pedig egymástól – többé-kevésbé – függetlenül viselkedő neuronokból állnak.

2.1.1. definíció (neuron). Egy $h : \mathbb{R}^d \rightarrow \mathbb{R}$ függvényt neuronnak nevezek, ha létezik $\mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}$, és $\phi : \mathbb{R} \rightarrow \mathbb{R}$ függvény, hogy

$$h(\mathbf{x}) = \phi(\mathbf{w}^\top \mathbf{x} + b) \tag{2.1}$$

\mathbf{w} a neuronhoz tartozó **súlyvektor**, ϕ pedig az **aktivációs függvény**. A b skalár a nemzetközi szakirodalomban használják a *bias* kifejezést.

2.1.2. definíció (réteg). Egy $\mathbf{h} : \mathbb{R}^{d_1} \rightarrow \mathbb{R}^{d_2}$ függvényt rétegnek nevezek, ha minden koordinátafüggvénye neuron.

Egy réteg tehát

$$\mathbf{h}(\mathbf{x}) = \bar{\phi}(\mathbf{W}\mathbf{x} + \mathbf{b}) \quad (2.2)$$

alakú, ahol $\mathbf{W} \in \mathbb{R}^{d_1 \times d_2}$, $\mathbf{b} \in \mathbb{R}^{d_2}$, és $\bar{\phi} : \mathbb{R}^{d_2} \rightarrow \mathbb{R}^{d_2}$ függvény.

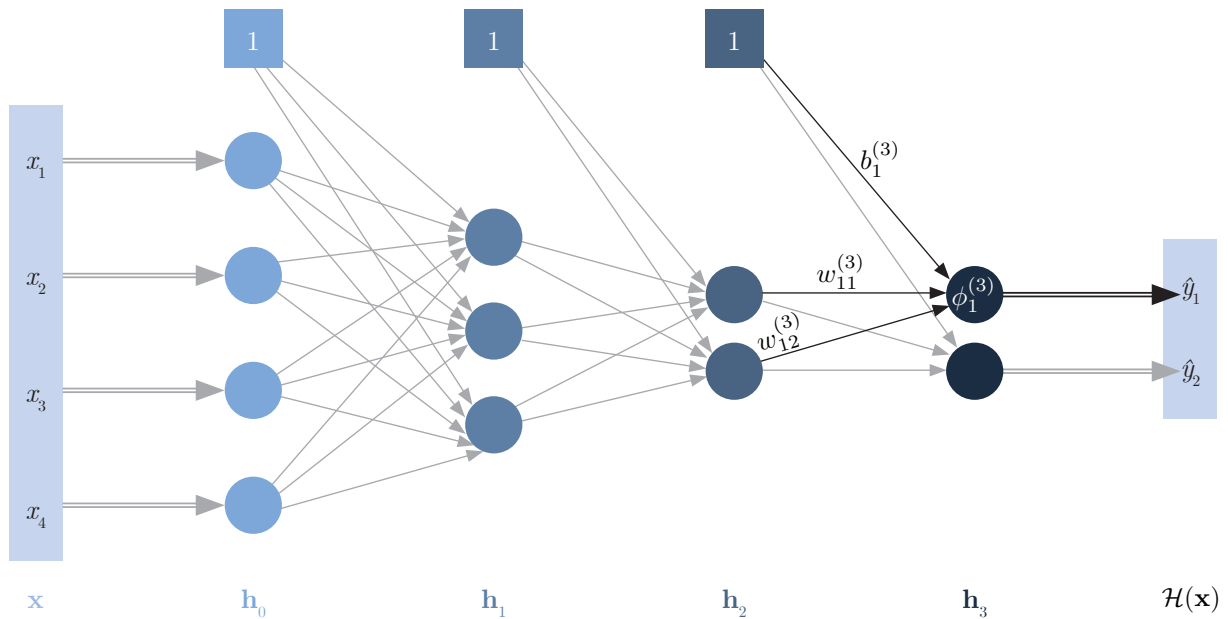
Ezek alapján megfogalmazható a mesterséges neuronháló egyik fontos fajtájának pontos definíciója.

2.1.3. definíció. $\mathcal{H} : \mathbb{R}^r \rightarrow \mathbb{R}^s$ egy előrecsatolt neurális háló, ha léteznek $m, d_0, d_1, \dots, d_m \in \mathbb{N}$ számok ($d_0 = r, d_m = s$) és $\mathbf{h}_i : \mathbb{R}^{d_{i-1}} \rightarrow \mathbb{R}^{d_i}$ ($1 \leq i \leq m$) (2.2) alakú rétegek, hogy

$$\mathcal{H} = \mathbf{h}_m \circ \mathbf{h}_{m-1} \circ \dots \circ \mathbf{h}_1 \quad (2.3)$$

m a rétegek száma, másképpen a neuronháló mélysége. $\max d_i$ a háló **szélessége**. A definícióhoz a gyakorlatban hozzá szokás venni egy $\mathbf{h}_0 = \text{id}$ **bemeneti réteget**. \mathbf{h}_m a **kimeneti réteg**, a többi réteg pedig **rejtett réteg**. Ezek koordinátafüggvényei a kimeneti, illetve rejtett neuronok.

A \mathcal{H} hálónak megfeleltethető egy irányított gráf, amelynek csúcsai a neuronoknak felelnek meg, az élek pedig azt reprezentálják, hogy az egyes neuronok a bemenetüket mely más neuronok kimenetéből kapják. Ekkor a rétegek független csúcshalmazok.



1. ábra. Egy $\mathcal{H} = \mathbf{h}_3 \circ \mathbf{h}_2 \circ \mathbf{h}_1$ előrecsatolt neurális háló gráfja, ahol $\mathbf{h}_i(\mathbf{x}_i) = \bar{\phi}^{(i)}(\mathbf{W}^{(i)}\mathbf{x}_i + \mathbf{b}^{(i)})$, $\bar{\phi}^{(i)} = (\phi_1^{(i)}, \dots, \phi_{d_i}^{(i)})$

A rétegek működésének fontos összetevője az aktivációs függvény megválasztása. A $\bar{\phi}$ aktivációs függvény legtöbbször egy $\phi : \mathbb{R} \rightarrow \mathbb{R}$ aktivációs függvény koordinátánkénti alkalmazása. A kézenfekvő választás a $\phi(x) = x$ úgynevezett **lineáris aktiváció** lenne. Könnyen látható azonban, hogy ha minden rétegben lineáris aktivációt választunk, akkor a neurális háló egyetlen affin leképezéssé egyszerűsödik, amelyre nem teljesül, hogy globálisan jól közelítene más függvényeket.

A gyakorlatban általában majdnem mindenhol differenciálható, monoton és korlátos aktivációs függvényeket alkalmaznak. A leggyakoribb választások (a lineáris aktiváció mellett):

- $\frac{1}{1+e^{-x}}$ (szigmoid aktiváció),
- $\tanh x$,
- $\max(0, x)$ (*rectified linear unit*, röviden ReLU),
- $\max(0, x) + a \min(0, x)$, $a \in (0, 1)$ paraméter (leaky ReLU).

Az approximálandó függvény gyakran egy eloszlásfüggvény lesz. Ilyen esetekben gyakori választás a kimeneti réteg aktivációs függvényeként a

$$\text{softmax}(\mathbf{x}) = \left(\frac{e^{x_j}}{\sum_{k=1}^q e^{x_k}} \right)_{j=1}^q \quad (2.4)$$

függvény, ennek értéke ugyanis mindig egy eloszlás lesz, azaz $\text{softmax}(\mathbf{x})$ koordinátái minden esetben nemnegatívak és az összegük 1.

Mivel a neurális háló fő célja függvények közelítése, természetes kérdés, hogy milyen függvényeket tudnak közelíteni, és milyen pontossággal. Nyilvánvalóan a megfelelően mély előre-csatolt háló elég bonyolult függvényeket fognak leírni. Azonban a rétegek számát és felépítését is erősen korlátozva még mindig erős közelítésekre alkalmas függvénycsaládot kapunk.

2.1.4. definíció (ϕ -háló). Legyen $\phi : \mathbb{R} \rightarrow \mathbb{R}$ függvény. Ekkor egy \mathcal{H} előre-csatolt neurális hálót ϕ -hálónak nevezek, ha minden rejtett neuronja ϕ aktivációs függvényű, a kimeneti neuronjai pedig lineárisak.

Ez az architektúra elég ahhoz, hogy egy folytonos függvényt tetszőlegesen jól közelítsen egy korlátos halmazon. Erről szólnak az úgynevezett univerzációs approximációs tételek.

2.1.5. tétel. Legyen $\phi : \mathbb{R} \rightarrow \mathbb{R}$ függvény, amely minden polinomtól eltér egy pozitív mértékű halmazon. Ekkor tetszőleges $K \subset \mathbb{R}^r$ kompakt halmaz esetén

1. ha ϕ folytonos, akkor a kétrétegű ϕ -háló sűrűek $\mathcal{C}(K, \mathbb{R}^s)$ -ben a szuprémumnorma szerint [8] és
2. ha ϕ mérhető és lokálisan L^p -beli, akkor a kétrétegű ϕ -háló sűrűek $L^p(K, \mathbb{R}^s)$ -ben ($1 \leq p < \infty$). [9]

A fenti két tételből látszik, hogy a kétrétegű ϕ -háló kompakt halmazon jól közelít függvényeket a leggyakrabban használt normák szerint. Fontos azonban, hogy a fenti tételek nem korlátozzák a háló szélességét.

A gyakorlatban széles háló helyett gyakran mély hálókat használnak, ezért ismertetek egy friss eredményt korlátos szélességű mély hálóról. Az elmúlt két évben megmutatták, hogy a fenti tételek némi módosítással kimondhatók akkor is, ha \mathcal{H} szélességét korlátozzuk, de a mélységét nem. (Sőt folytonos függvények egyenes közelítéséhez még azt sem kell feltenni, hogy ϕ nem polinom.)

2.1.6. tétel. Legyen $\phi : \mathbb{R} \rightarrow \mathbb{R}$ folytonos függvény, amely legalább egy pontban folytonosan differenciálható nemnulla deriválttal. Ekkor tetszőleges $K \subset \mathbb{R}^r$ kompakt halmaz esetén

1. ha ϕ nem affin leképezés, akkor az $r + s + 2$ széles háló sűrűek $\mathcal{C}(K, \mathbb{R}^s)$ -ben a szuprémumnorma szerint [10] és
2. ha ϕ nem polinomfüggvény, akkor a $\max(r+2, s+1)$ széles ϕ -háló sűrűek $L^p(K, \mathbb{R}^s)$ -ben ($1 \leq p < \infty$). [11]

2.2. Tanulás

Az univerzális approximációs tételek tehát biztosítják megfelelő feltételek mellett a neurális hálók mint függvényközelítések erősségét. Kérdés, hogy egy adott F függvényhez hogyan található meg az őt legjobban közelítő háló.

A gyakorlatban emellett általában F sem ismert, csak F értékei egy – általában véges – \mathcal{D} adathalmazon. Legyen F például az a függvény, amely egy 256×256 -os RGB képhez 1-et rendel, ha a kép macskát ábrázol, és 0-t, ha kutyát. Egy ilyen feladat megoldásához általában a rendelkezésünkre áll néhány kép, amelyekről tudjuk, hogy kutyákat vagy macskákat ábrázolnak-e, és ez alapján keresünk egy olyan hálót, amely bármilyen, kutyát vagy macskát ábrázoló képet képes besorolni a megfelelő osztályba. (Ez a feladat az úgynevezett képklasszifikáció egy esete, amelyről a 3. fejezetben részletesebben lesz szó.) Az ötlet az, hogy keressünk egy olyan \mathcal{H} hálót, amely ezen a \mathcal{D} adathalmazon jól approximálja F -et, és ezt lehetőleg olyan módszerrel tegyük, hogy az így kapott háló \mathcal{D} -n kívüli pontokban is jó közelítést adjon.

Ehhez az ismert adatpontok és esetleg F feltételezett tulajdonságai alapján kiválasztunk egy hálóarchitektúrát, azaz lerögzítjük a háló mélységét, rétegeinek szélességét és az aktivációs függvényeket, de a súly és bias paramétereket nem. Ekkor a hálót $\mathcal{H}(\Theta, \cdot)$ alakban keressük, ahol a $\Theta \in \mathbb{R}^P$ paraméterek a súlymátrixok és biasvektorok koordinátái.

Megjegyzés. Nem kell hogy P megegyezzen a súlymátrixok és biasvektorok dimenzióinak összegével. A gyakorlatban gyakran kevesebb, mert bizonyos bias- vagy súlyértékeket rögzítenek (általában 0-ra), vagy olyan hálókat keresnek, ahol egyes neuronok bizonyos súlyai megegyeznek.

Miután eldöntöttük, hogy pontosan milyen alakú \mathcal{H} hálókat keresünk, és van egy Θ paraméterezésünk, kérdés, hogy F és $\mathcal{H}(\Theta, \cdot)$ eltérését hogyan mérjük.

2.2.1. definíció (veszteségfüggvény). Egy $\mathcal{L} : \text{Im } F \times \text{Im } \mathcal{H} \rightarrow \mathbb{R}$ függvény veszteségfüggvénye (*loss function*) az F -et approximáló $\mathcal{H}(\Theta, \cdot)$ (2.3) alakú előrecsatolt hálóknak, ha \mathcal{L} nemnegatív, és minden \mathbf{y} -ra $\mathcal{L}(\mathbf{y}, \mathbf{y}) = 0$.

Ekkor ha adott egy \mathcal{L} veszteségfüggvény, akkor keressük az

$$\operatorname{argmin}_{\Theta \in \mathbb{R}^P} \sum_{\mathbf{x} \in \mathcal{D}} \mathcal{L}(F(\mathbf{x}), \mathcal{H}(\Theta, \mathbf{x})) \quad (2.5)$$

értéket. A $\mathcal{H}(\Theta, \mathbf{x})$ háló tanításának nevezünk minden olyan iteratív algoritmust, amely minden iterációjában a fenti összeget próbálja javítani Θ -nak valamilyen optimalizációs szabály szerinti módosításával.

Az aktivációs függvényekhez hasonlóan a veszteségfüggvény megválasztása is fontos szerepet játszik a háló működésében. A legkézenfekvőbb választás itt talán a négyzetátlag veszteségfüggvény: $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{s} \|\mathbf{y} - \hat{\mathbf{y}}\|^2$. Ennek hátránya, hogy kis értékeknél nagyon gyorsan eltűnik, ezért nem megfelelő például softmax kimeneti aktiváció esetén, ahol $\mathbf{0} < \hat{\mathbf{y}} < \mathbf{1}$ teljesül. Mivel a softmax aktivációt általában olyan esetekben használják, amikor a megtanulni kívánt hozzárendelés one-hot vektorokból, azaz egyetlen 1-es és mindenhol máshol 0-t tartalmazó vektorokból áll, lehet egy speciális alakú \mathcal{L} -et definiálni.

2.2.2. definíció (categorical crossentropy). Legyen $\mathbf{y} \in \{0, 1\}^s$ egységvektor. Ekkor \mathcal{L} categorical crossentropy veszteségfüggvény, ha

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = -\log \hat{y}_j \quad (2.6)$$

Tegyük fel, hogy \mathcal{L} egy differenciálható függvény. (Ezt teljesíti mind a négyzetátlag, mind a categorical crossentropy.) Legyen Θ egy adott paraméterezés, $\mathbf{x} \in \mathcal{D}$, $\mathbf{y} = F(\mathbf{x})$ és $\hat{\mathbf{y}} = \mathcal{H}(\Theta, \mathbf{x})$. (Tehát $\hat{\mathbf{y}}$ valójában függ Θ -tól.) Ekkor a $\frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \Theta}$ deriváltvektor megmutatja, hogy \mathcal{L} lokálisan milyen irányban változik a leggyorsabban.

Megjegyzés. Az egyik leggyakrabban használt aktivációs függvény, a ReLU nem differenciálható minden pontjában, ezért ilyen aktivációk esetén $\frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \Theta}$ sem létezik minden pontban. Ez azonban a gyakorlatban nem jelent gondot, ha a függvény majdnem mindenhol differenciálható, mert úgylis csak véges pontossággal tudunk számolni.

A következőkben bemutatok két gyakori optimalizációs módszert. Ezekben végig azt fogom feltételezni, hogy Θ módosításához rendelkezésre állnak az approximálandó függvény $\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \dots, \mathbf{y}^{(n)}$ értékei és a háló $\hat{\mathbf{y}}^{(1)}, \hat{\mathbf{y}}^{(2)}, \dots, \hat{\mathbf{y}}^{(n)}$ közelítései.

Gradient descent (gradiensleereszkedés)

Ez a módszer azt a megfigyelést veszi alapul, hogy a veszteségfüggvény Θ szerinti parciális deriváltja megmutatja, hogy (Θ, \mathbf{x}) egy infinitezimális környezetében $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})$ milyen irányban nő a leggyorsabban. Ezért a gradient descent módszer iterációs szabálya

$$\Theta \leftarrow \Theta - \eta \left(\frac{1}{n} \sum_{i=1}^n \frac{\partial \mathcal{L}(\mathbf{y}^{(i)}, \hat{\mathbf{y}}^{(i)})}{\partial \Theta} \right) \quad (2.7)$$

Itt η a tanulási ráta (*learning rate*), szinte mindig egy $(0, 1)$ -beli szám. Gyakori választás az $\eta = 10^{-2}$ vagy $\eta = 10^{-3}$.

Megjegyzés. Az $n = 1$ esetet, amikor a gradienst egyetlen alappont alapján számolják ki, **sztochasztikus gradiensleereszkedés** (*stochastic gradient descent*, SGD) módszernek nevezik. Ezt az elnevezést néha akkor is alkalmazzák, ha $n \neq 1$, de kicsi az adathalmaz méretéhez viszonyítva.

A gyakorlatban bizonyos adathalmazokon a $\frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \Theta}$ deriváltak iránya jelentősen változik iterációnként, ezért a lépések effektív mérete igen kicsi lesz. A legtöbb, gyakorlatban használt módszer a gradient descent módszert veszi alapul, a módosítások pedig azt próbálják elérni, hogy az iterációnkénti oszcilláció csökkenjen.

Gradient descent momentummal

A gradient descent egyik gyakori módosítása egy $\gamma \in (0, 1)$ lendületi vagy momentumparaméter bevezetése. Ekkor az iterációs szabály

$$\mathbf{V} \leftarrow \gamma \mathbf{V} + \eta \left(\frac{1}{n} \sum_{i=1}^n \frac{\partial \mathcal{L}(\mathbf{y}^{(i)}, \hat{\mathbf{y}}^{(i)})}{\partial \Theta} \right), \quad \Theta \leftarrow \Theta - \mathbf{V} \quad (2.8)$$

(\mathbf{V} kezdőértéke $\mathbf{0}$.)

Ez a módszer segít elérni, hogy a gradiensek iránya kevesebbet változzon. $\gamma = \frac{9}{10}$ egy gyakori választás. (A $\gamma = 0$ választás lenne az egyszerű gradient descent.)

A fenti két módszeren kívül van még néhány elterjedt optimalizációs szabály, mint az RMSProp vagy az Adam, amelyek minden $\vartheta \in \Theta$ paramétert $\frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \vartheta}$ -tól nemlineárisan függő módon frissítenek.

A fenti képletekben az n értéket szokás (mini)batchméretnek nevezni. Egy **minibatch** azon adatpontok halmaza, amelyeket a háló egyszerre értékel ki, mielőtt elvégezné az optimalizációs lépést. Egy batch lehet az egész \mathcal{D} adathalmaz, de gyakrabban annak $n < |\mathcal{D}|$ méretű részhalmazait veszik, ahol n -et számítási okokból érdemes 2-hatványinak választani. (32, 64, 128 és 256 gyakori választások.) Az a ciklus, amely alatt az algoritmus feldolgozza az összes adatpontot, egy **epoch**.

A batchméret, tanulási ráta és a háló tanításának más paraméterei a **hiperparaméterek**. Ez az elnevezés arra szolgál, hogy megkülönböztesse őket Θ elemeitől, a **tanulható paraméterektől**.

Paraméterinicializáció

Egy háló tanítása tehát úgy néz ki, hogy valahány epochon keresztül veszi az algoritmus a \mathcal{D} adathalmazt, azt felbontja batchekre, majd minden batchen kiszámolja a veszteségfüggvényt, és frissíti a paramétereket a veszteségfüggvény alapján. Felmerül a kérdés, hogy milyen kezdeti értéket érdemes adni Θ -nak.

A természetes válasz a $\Theta = \mathbf{0}$ kezdőérték lenne, ez azonban több problémához vezet a gyakorlatban. A 0-ra inicializálás működik a biasvektorok esetében, de a súlyokat érdemesebb véletlenszerűen inicializálni. Vegyük egy $h : \mathbb{R}^d \rightarrow \mathbb{R}$ neuron $w \in \mathbb{R}$ súlyát. (Itt d azon bemenetek száma, amelyekről függ h értéke. Tehát ha egy h -hoz tartozó súly 0 és nem tanulható, akkor azt nem számolom hozzá d értékéhez.)

Ekkor kezdeti értéként $w \sim \mathcal{N}(0, \delta^2)$ vagy $w \sim \mathcal{U}(-\delta, \delta)$ szokásos választás, ahol δ lehet kis konstans (például 10^{-2}), de értéke függhet d -től is. (Szokásosan ez lehet $\frac{1}{\sqrt{d}}$.)

Szofisztikáltabb inicializációs módszereknél δ értéke attól is függ, hogy h bemenetként hány másik neuronhoz csatlakozik. Az úgynevezett Xavier-inicializációnál például ℓ kimeneti kapcsolat esetén δ arányos $\frac{1}{\sqrt{\ell+d}}$ -vel.

Közlök itt egy tételt, amely bizonyos típusú hálók konvergenciáját biztosítja a tanított adatpontokon.

2.2.3. tétel. Tegyük fel, hogy \mathcal{L} akárhányszor deriválható, és minden deriváltja Lipschitz-tulajdonságú közös Lipschitz-konstanssal. Legyen $\mathcal{H}(\Theta, \cdot) : \mathbb{R}^r \rightarrow \mathbb{R}^s$ egy minden rejtett rétegében d széles háló ReLU rejtett aktivációkkal és lineáris kimeneti aktivációval, ahol Θ a rejtett rétegek súlyait paraméterezi, az első kivételével. Legyenek a rejtett rétegek súlyainak inicializációi $\mathcal{N}(0, \frac{2}{d})$, a kimeneti rétegének $\mathcal{N}(0, \frac{1}{d})$ eloszlásúak. Legyen \mathcal{D} egy véges adathalmaz és $F : \mathbb{R}^r \rightarrow \mathbb{R}^s$ tetszőleges függvény.

Ekkor tetszőleges $\varepsilon > 0$ mellett létezik $D, N \in \mathbb{N}$ és $\pi : \mathbb{N} \rightarrow [0, 1]$, hogy $d \geq D$ esetén az η tanulási rátájú gradient descent algoritmus legalább $\pi(d)$ valószínűséggel N iteráció alatt olyan Θ paraméterezést talál, hogy

$$\sum_{\mathbf{x} \in \mathcal{D}} \mathcal{L}(F(\mathbf{x}), \mathcal{H}(\Theta, \mathbf{x})) < \varepsilon$$

teljesül valamilyen η -ra, ahol $\pi(d) \rightarrow 1$. [12]

2.3. Mély neuronháló tanításának nehézségei

Az alábbiakban felsorolok néhány gyakori jelenséget, amelyek megnehezítik mély (általában több mint három-négy rejtett réteggel rendelkező) neuronháló tanítását, illetve mutatok néhány elterjedt megoldást a problémákra. Az itt leírtak részben a [13] könyvön alapulnak.

Eltűnő és felrobbanó gradiensek

Tekintsünk egy (2.3) alakú \mathcal{H} előrecsatolt hálót. Ez egy sokszorosán összetett függvény, amelynek (2.2) szerint leírt rétegei szintén összetett függvények. Így az $\mathcal{L}(\mathbf{y}, \cdot) \circ \mathcal{H}(\cdot, \mathbf{x})$ függvény deriváltja (amely a gradiensalapú optimalizációs módszerekhez szükséges) kiszámítható a láncszabály segítségével.

Ebben a deriváltban a $\frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \mathbf{y}}$ parciális deriválton kívül megjelennek a rétegek és lényeges módon az aktivációs függvények deriváltjai. A leggyakrabban használt aktivációs függvények (a ReLU-n és annak változatain kívül) deriváltja minden pontban $(0, 1)$ -beli, ezért sok ilyen összeszorozva a $\frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \Theta}$ gradiens nagyon kicsi lesz, „eltűnik”, és a gradiensalapú optimalizációs módszerek nem fogják megtalálni a globális minimumhelyet.

A legegyszerűbb válasz erre az, ha a háló paraméterei nagy kezdeti értékeket kapnak, ekkor azonban a derivált kezelhetetlenül nagy lesz, „felrobban”. A gyakorlati megoldás erre a problémára elővigyázatos paraméterinicializáció, normalizációs rétegek bevezetése, illetve speciális architektúrájú háló keresése, ahol ezek a gondok nem lépnek fel. (Ilyenek a konvolúciós és reziduális háló, amelyeket a későbbiekben bemutatok.)

Túltanulás

A tanulással kapcsolatban eddig végig a \mathcal{D} tanító adathalmazról volt szó. A gyakorlatban azonban sosem az a cél, hogy a \mathcal{D} -hez tartozó kimenet-bemenet párokat megtanulja a háló, hanem hogy ezek alapján jól tudjon általánosítani más, ismeretlen értékekre. A tanítás során ezért a rendelkezésre álló adatok egy részét általában félreteszik, és azokon nem tanítják a hálót, csupán minden epochban megméri rajtuk a teljesítményt. Előfordulhat, hogy a tanító adathalmazon a háló már nagyon jól közelíti a pontos értékeket, de a validációs adatokon egyre rosszabb eredményt ér el, azaz egyre rosszabbul általánosít. Ez a jelenség a túltanulás (*overfitting*).

A túltanulás gyakori oka, hogy a hálónak valamivel több paramétere van, mint amennyi szükséges ahhoz, hogy megtanulja \mathcal{D} -t, és ezért a tanító adathalmaz jelentéktelen részleteire kezd el rátanulni. A következőkben bemutatott módszereknek az a céljuk, hogy ezt a jelenséget megpróbálják korlátozni.

Tyihonov-regularizáció

A túl sok paramétert lehetséges úgy kiküszöbölni, ha az optimalizációs algoritmus bünteti Θ nagy paramétereit. Az L^2 - vagy Tyihonov-regularizáció ehhez bevezeti az

$$\tilde{\mathcal{L}}(\Theta) = \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) + \nu \|\Theta\|^2 \quad (2.9)$$

értéket ($\nu > 0$). Ekkor az $\tilde{\mathcal{L}}$ veszteségfüggvényen elvégzett optimalizáció várhatóan kisebb Θ értéket fog találni. Ezt a fajta regularizációt szokás súlyelhalásnak (*weight decay*) nevezni, mert az optimalizáció hajlamos csökkenteni a súlyok abszolút értékét.

Megjegyzés. A *weight decay* kifejezést arra is használják, amikor a gradiensalapú módszereknél a súlyok nem $\Theta \leftarrow \Theta - \mathbf{V}$, hanem $\Theta \leftarrow (1 - \nu)\Theta - \mathbf{V}$ szabály szerint módosulnak. Ez ekvivalens a Tyihonov-regularizációval sima gradient descent esetén, de más optimalizációs módszerek esetén nem az. [14]

Batchnormalizáció

Egy másik lehetőség a felrobbanó és eltűnő gradiensok problémájának megszüntetésére, ha csökkentjük a rétegek változó súlyai okozta „zajt”. Ennek egyik gyakori módszere a háló architektúrájának módosítása batchnormalizációval. A batchnormalizáció (röviden batchnorm) egy adott neuron kimenetét transzformálja batchenként úgy, hogy a batch kimenete τ várható értékű és ξ szórású legyen, ahol τ és ξ tanulható paraméterek.

A batchnormalizáció alkalmazható a neuron kimenetére, ekkor posztaktivációs batchnormalizációról beszélünk, de alkalmazható az aktivációs függvény előtt, ekkor preaktivációs batchnormalizációról beszélünk.

Ebben a fejezetben tehát áttekintettem a mesterséges neuronhálókra vonatkozó néhány általános tudnivalót. Mint említettem, a hálóarchitektúra választására befolyással van az, hogy milyen feladatot kell megoldani. A következő fejezet ezt fogja vizsgálni abban az esetben, amikor a feladat képek felismerése.

3. fejezet

Képklasszifikáció mély neuronhálókkal

A mesterséges neuronháló alkalmazásának egyik legjobban kutatott területe a gépi látás (*computer vision*) és azon belül is a képklasszifikációs feladatok. Képklasszifikáció során adott osztályoknak egy halmaza, és a feladat az, hogy a bemeneti képekről megmondjuk, melyik osztályba tartoznak.

Képklasszifikáció esetén a háló bemenete hagyományosan a kép valamilyen – általában RGB – reprezentációja, azaz a kép pixeleinek egy listája, ahol minden pixelt három, 0 és 255 közötti egész szám ír le, amelyek a pixelben a vörös, zöld és kék színek intenzitását kódolják el. A kimenet azt mutatja, hogy a kép melyik osztályba tartozik. Ennek reprezentációja N osztály esetén szokásosan egy $[0, 1]^N$ -beli vektor, amelynek az a koordinátája 1, amelyik a képen ábrázolt osztályhoz tartozik, a többi 0. Emiatt a reprezentáció miatt nagyon gyakori képklasszifikációnál a softmax kimeneti aktivációk használata. Ebben az esetben a kimenet felfogható úgy, hogy minden koordinátája azt írja le, hogy a háló szerint mekkora valószínűséggel tartozik a bemeneti kép az adott osztályba.

A háló predikciója ekkor az az osztály, amelybe a legnagyobb valószínűséggel tartozik szerinte a kép, egy adott adathalmazon mért **pontossága** pedig az eltalált bemenetek száma osztva az összes bemenet számával.

3.1. Képek mint bemeneti adatok

Gyakran használt adathalmazok

Az alábbiakban ismertetek néhány, a nyilvánosság számára elérhető adathalmazt. Az ezeken végzett mérések alkalmasak arra, hogy különböző modellek pontosságát össze lehessen hasonlítani.

A Canadian Institute for Advanced Research-ről elnevezett **CIFAR** adathalmazok 32×32 -es fényképekből állnak. A CIFAR-10 tíz osztályból tartalmaz osztályonként hatezer képet, míg a CIFAR-100 száz osztályból osztályonként hatszázat. Mivel ezen adathalmazok viszonylag kis méretűek, jól használhatók olyan esetekben, amikor nincsenek meg a megfelelő erőforrások egy nagyobb adathalmazhoz. [15]

Az **ImageNet** egy több mint tizennégymillió képből álló adathalmaz. Az ImageNet Large Scale Visual Recognition Challenge (ILSVRC) egy több egymást követő évben megrendezett verseny az ImageNet adathalmaz egy részén, amely több mint 1,2 millió

képből áll, ezer kategóriából véve. A képek felbontása változó, átlagos méretük 400×350 pixel körüli. Az irodalomban az ImageNet nevet általában egy adott évi ILSVRC adathalmazára használják. [16, 17]

Megjegyzés. Az ImageNet adathalmaz és a benne lévő osztályok számának mérete miatt egy-egy kép akár több osztályból származó objektumokat is ábrázolhat, amely nagyban megnehezíti a képklasszifikációt. Ezért az ILSVRC versenyeken a modelleknek megengedett, hogy egy képre öt különböző kimenetet adjanak, és a modell predikciója akkor számít helyesnek, ha az öt kimenet közül az egyik megegyezik a kép címkéjével.

Előfeldolgozás

A természetes képek klasszifikációjának fontos része az előfeldolgozás. Előfeldolgozás alatt az adathalmaz minden olyan transzformációját értem, amely a tanítás megkezdése előtt történik, és célja az, hogy a háló könnyebben tudja kezelni, illetve megtanulni az adatokat. RGB képeknél a leggyakoribb előfeldolgozási eljárás a pixelintenzitások normalizálása, hogy a bemenet minden koordinátája a $[0, 1]$, esetleg $[-1, 1]$ intervallumba essék. Ez azért hasznos, mert az egyes bemeneti értékek közti túl nagy eltérés ront a gradiensalapú módszerek hatékonyságán. [18]

Szintén gyakori előfeldolgozási lépés a globális kontrasztnormalizáció (*global contrast normalisation*, röviden GCN). [19]

3.1.1. definíció (kontraszt). Legyen $\mathbf{x} \in \mathbb{R}^{\ell \times v \times 3}$ képi adat (ahol ℓ a kép magassága, v a szélessége, és a x_{ijr} az (i, j) helyen lévő pixelben az r . színcsatorna intenzitása). Ekkor \mathbf{x} kontrasztjának a

$$\kappa(\mathbf{x}) = \sqrt{\frac{1}{3\ell v} \sum_{i=1}^{\ell} \sum_{j=1}^v \sum_{r=1}^3 (x_{ijr} - \bar{\mathbf{x}})^2} \quad (3.1)$$

mennyiséget nevezem, ahol

$$\bar{\mathbf{x}} = \frac{1}{3\ell v} \sum_{i=1}^{\ell} \sum_{j=1}^v \sum_{r=1}^3 x_{ijk}$$

3.1.2. definíció. \mathbf{x} globális kontrasztnormalizáltja $\tilde{\mathbf{x}}$, ahol

$$\tilde{x}_{ijr} = S \cdot \frac{x_{ijr} - \bar{\mathbf{x}}}{\max\left(\varepsilon, \sqrt{\lambda + \kappa^2(\mathbf{x})}\right)} \quad (3.2)$$

($\varepsilon, \lambda > 0$.)

Megjegyzés. Az ε és λ értékek szerepe, hogy 0 kontrasztú képeket is lehessen normalizálni. Ezért az egyik értéket (általában λ -t) 0-ra szokás beállítani. S szokásos értéke 1.

Adataugmentáció

A mesterséges neuronháló általános problémája az, hogy a tanításukhoz nagyon nagy mennyiségű adatra van szükség, amely gyakran nem áll rendelkezésre. Ilyenkor érdemes adataugmentációt használni a tanító adathalmazon. Augmentáció során az adathalmaz méretét mesterségesen megnöveljük újabb képek hozzáadásával, amelyeket az eredetiktől

generálunk valamelyik transzformáció segítségével. Gyakori augmentációs módszer forgatások, tükrözések alkalmazása, illetve a bemeneti képekből véletlenszerűen valamikora rész kivágása. (Ez a lépés elengedhetetlen akkor, ha az adatpontok mérete változó, mint például az imagenetes képek esetében.)

Az augmentáció hatékonysága azon múlik, hogy a megfelelően kiválasztott transzformációk nem változtatják meg a bemenet lényegi tulajdonságait. (Egy macskát ábrázoló képet a függőleges tengelyre tükrözve még mindig nyilvánvalóan egy macskát ábrázoló képet kapunk.)

3.2. Konvolúciós hálók

Az elmúlt években a képklasszifikáció legerősebb eszközei a különböző konvolúciós neurális hálók (*convolutional neural networks*, avagy CNN-ek) voltak. Ezek a modellek a múlt évtized közepére kevesebb mint 4%-os hibával teljesítettek az ImageNet adatbázison, amely jobb mint az 5,1%-os emberi teljesítmény. [17]

A konvolúciós hálók fő ereje abban rejlik, hogy kihasználják a bemeneti pixelek térbeli elhelyezkedését. Ezért az egyes rejtett rétegek kimenetére érdemes úgy gondolni, mint háromdimenziós pontthalmazokra, és így van szélességük, magasságuk és mélységük. A **mélység** itt a csatornák számát jelenti, nem összekeverendő a háló mélységével, amely a rétegek száma. Így a bemeneti réteg csatornái megfelelnek a piros, zöld és kék színcsatornáknak.

A konvolúciós hálók által használt két legfontosabb eszköz a konvolúciós és a pooling réteg.

Konvolúciós réteg

3.2.1. definíció. Legyenek $\ell, v, c, \ell', v', c', p_\ell, k_\ell, p_v, k_v$ és s természetes számok, hogy

$$\ell' = \left\lfloor \frac{\ell + 2p_\ell - k_\ell}{s} \right\rfloor + 1, \quad v' = \left\lfloor \frac{v + 2p_v - k_v}{s} \right\rfloor + 1$$

teljesül. Legyen $\mathbf{h} : \mathbb{R}^{\ell \times v \times c} \rightarrow \mathbb{R}^{\ell' \times v' \times c'}$ réteg (2.2) szerint, és legyen $\mathbf{h}(\mathbf{x}) = \mathbf{x}'$, ahol

$$x'_{mnq} = \phi \left(\sum_{i=1}^{k_\ell} \sum_{j=1}^{k_v} \sum_{r=1}^c w_{ijr}^{(q)} x_{s(m-1)-p_\ell+i, s(n-1)-p_v+j, r} + b^{(q)} \right) \quad (3.3)$$

ahol ϕ a ReLU aktivációs függvény, $w_{ijr}^{(q)}, b^{(q)} \in \mathbb{R}$ és $x_{\alpha\beta r} = 0$, ha $\alpha \leq 0$, $\alpha > \ell$, $\beta \leq 0$ és $\beta > v$ közül bármelyik teljesül. Ekkor \mathbf{h} konvolúciós réteg.

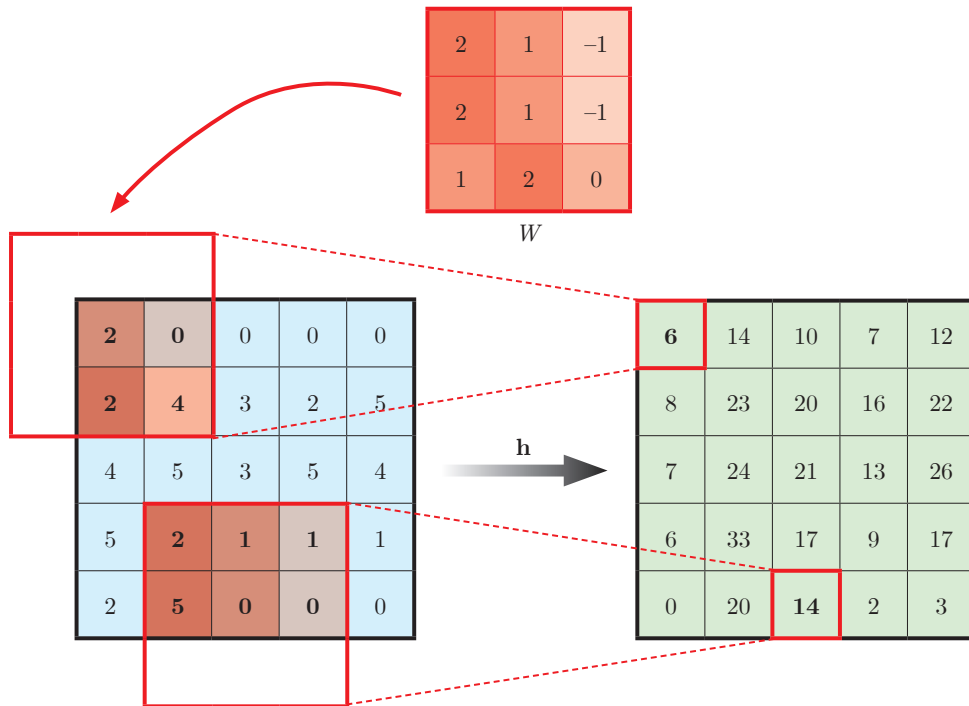
Megjegyzés. ReLU helyett természetesen bármilyen más aktivációs függvény is használható, azonban a gyakorlatban a konvolúciós hálóknak szinte kizárólag ReLU-t használnak.

Mit is jelent pontosan ez a definíció? Először tegyük fel, hogy $c' = 1$, $p_\ell = p_v = 0$, $s = 1$, $b^{(1)} = 0$, és tekintsünk el a ϕ függvénytől. Ekkor a konvolúciós réteg felfogható mint egyetlen $W \in \mathbb{R}^{k_\ell \times k_v \times c}$ **filter**. Ekkor a konvolúciós réteg alkalmazása annyit tesz, hogy az \mathbf{x} bemeneten „végigcsúsztatjuk” ezt a W filtert, és minden állásnál összeadjuk az

éppen lefedett $k_\ell \cdot k_v \cdot d$ számot W koordinátaival súlyozva. A $b^{(1)}$ bias szerepe csupán annyi, hogy a kimenet minden koordinátáját eltoljuk egy konstanssal, ϕ pedig a negatív koordinátákat 0-ra állítja. Amennyiben $c' \geq 1$, akkor több ilyen filtert alkalmazunk, a kimenet mind a c' csatornájához egyet.

Megjegyzés. A konvolúciós réteg tanulható paraméterei a $w_{ijr}^{(q)}$, esetleg a $b^{(q)}$ értékek, így összesen $(k_\ell k_v + 1)c'$ érték, illetve $k_\ell k_v c'$, ha a b biasok értéke konstans. (A konstans 0 választás gyakori a konvolúciós hálók esetében.) Tehát ha a bemenet egy 32×32 -es kép három színcsatornával, és $k_\ell = k_v = 3$, akkor a kimenet 30×30 -as méretű. Ha egy $\mathbf{h} : \mathbb{R}^{32 \times 32 \times 3} \rightarrow \mathbb{R}^{30 \times 30 \times 64}$ réteg minden súlya egymástól függetlenül tanulható, akkor a hálónak csak ebben a rétegben 176 947 200 súlyt kell megtanulnia. Ezzel szemben egy konvolúciós réteg csupán 1728 tanulható súlyt tartalmaz. A súlyoknak viszonylag kis száma miatt lehetséges nagyon mély konvolúciós hálók építése is.

A p_ℓ és p_v párnázási vagy **padding** paraméterek általában arra szolgálnak, hogy $\ell' = \ell$, $v' = v$ teljesüljön. Érdekes rájuk úgy gondolni, mintha a bemeneti képet mindkét oldalán „kipárnáznánk” p_ℓ , illetve p_v mennyiségű 0-val.



2. ábra. A \mathbf{h} konvolúciós réteg bemenete, valamint egy csatornájához tartozó filter és kimenet, nulla bias, $p_\ell = p_v = 1$ és $c = 1$ mellett.

Előfordulhat az is, hogy jelentősen csökkenteni akarjuk a bemenet méretét. Erre való az s paraméter, amelynek neve **lépésköz** (*stride*). $s > 1$ esetén a W filtert nem helyezzük rá a bemenetre minden lehetséges módon, csak s -esével lépkedve.

A gyakorlatban szinte mindig négyzet alakú képekkel dolgoznak minden rétegben, így $\ell = v$, $k_\ell = k_v$, $p_\ell = p_v$ teljesül. A lépésközre a leggyakoribb választás az 1, ritkán alkalmaznak 2, még ritkábban nagyobb lépésközt. $s = 1$ esetén a padding méretét szinte mindig akkorára választják, hogy a bemenet és kimenet mérete megegyezzen. A kimeneti csatornák d' számának általában 2-hatványt választanak. A filter leggyakoribb méretei 3×3 és 5×5 , de 1×1 -es konvolúciós rétegeket is használnak.

Pooling réteg

3.2.2. definíció. Legyenek $\ell, v, c, \ell', v', k_\ell, k_v$ és s természetes számok, hogy

$$\ell' = \frac{\ell - k_\ell}{s} + 1, \quad v' = \frac{v - k_v}{s} + 1$$

teljesül. Legyen $\mathbf{h} : \mathbb{R}^{\ell \times v \times c} \rightarrow \mathbb{R}^{\ell' \times v' \times c}$, $\mathbf{x} \mapsto \mathbf{x}'$, ahol

$$x'_{mnr} = \mu \left(\begin{bmatrix} x_{s(m-1)+1, s(n-1)+1, r} & \cdots & x_{s(m-1)+1, s(n-1)+k_v, r} \\ \vdots & \ddots & \vdots \\ x_{s(m-1)+k_\ell, s(n-1)+1, r} & \cdots & x_{s(m-1)+k_\ell, s(n-1)+k_v, r} \end{bmatrix} \right) \quad (3.4)$$

ahol $\mu : \mathbb{R}^{k_\ell \times k_v} \rightarrow \mathbb{R}$ valamilyen középértékfüggvény. Ekkor \mathbf{h} egy pooling réteg.

A pooling tehát hasonlít a konvolúcióhoz, ugyanakkor van jó pár fontos különbség. Egyrészt a pooling rétegeknek nincsenek tanulható paraméterei. Másrészt a pooling az egyes csatornákon egymástól függetlenül történik, így nem is változik a csatornák száma. Harmadrészt nem szokás padding alkalmazása.

Pooling esetén az s lépésköz általában 1-nél nagyobb, $s = 2$ és $k_\ell = k_v = 2$ gyakori választás. A konvolúcióhoz hasonlóan itt is $k_\ell = k_v$ teljesül általában. μ -re a két leggyakoribb választás a maximumfüggvény (*max pooling*), illetve a számtani közép (*average pooling*).

Mind a konvolúciós, mind a pooling réteg jól kezeli a bemenet térbeli transzformációit: ha ugyanazok a képpontok megjelennek a bemenet egy más részén, akkor a kimenet megfelelő részén is ugyanazok az értékek fognak megjelenni. Ez az egyik oka annak, hogy jól teljesítenek képklasszifikációs feladatoknál, hiszen képklasszifikáció esetén nem lényeges, hogy a keresett objektum a kép melyik részén van.

A legegyszerűbb konvolúciós hálókból néhány konvolúciós réteget követ egy pooling réteg, majd az utolsó poolingot a kimeneti réteghez egy **sűrű** (*fully connected*, néha FC) réteg kapcsolja, amelynek minden súlya tanítható. Egyes hálóknál több *fully connected* réteget használnak. Pusztán ezekkel a rétegekkel létrehozhatók több mint tíz réteg mély hálóknak, amelyek rendre jobban teljesítettek, mint a sekélyebb architektúrák.

Megjegyzés. Másokban a *fully connected* réteget megelőzi, esetleg helyettesíti is egy globális pooling réteg, amely egy $\mathbb{R}^{\ell \times v \times c} \rightarrow \mathbb{R}^c$ pooling függvény. (Azaz minden $\ell \times v$ -s csatornához egyetlen számot rendel.) A globális pooling réteg szinte mindig számtani közepet használ mint pooling operáció. [20]

Mivel ezek a mélyebb hálóknak jobb teljesítményt mutatnak, felmerül a kérdés, hogy érdemes-e még mélyebb modelleket építeni. Ez azonban nem megoldható pusztán még több réteg egymás mögé tételével; a tapasztalat azt mutatja, hogy az így keletkező modellek nem érnek el jobb teljesítményt. Erre nyújtanak egy lehetséges megoldást a reziduális hálóknak.

3.3. Reziduális hálóknak

2015-ben He et al. egy újfajta konvolúciós hálót hoztak létre, amelyet ResNetnek kereszteltek el, a *residual network* (reziduális háló) rövidítésékként. Ezzel a hálóval megnyerték a 2015-ös ILSVRC versenyt, majd eredményeiket a [21] cikkben jelentették meg. Később

a [22] cikkben közölték az eredeti ResNetnek egy módosított változatát. Ebben a részben ezen cikkek alapján ismertetem a ResNet hálókat.

Mint azt az előző részben már megjegyeztem, a tapasztalat azt mutatja, hogy a túl mély konvolúciós hálók teljesítménye rosszabb, mint sekélyebb megfelelőiké. Ennek nyilvánvalóan nem az az oka, hogy a mélyebb hálók kifejezőképessége rosszabb lenne. Ha ugyanis $\mathcal{H}(\Theta, \cdot)$ egy neurális háló (az egyszerűség kedvéért most lineáris aktivációjú kimeneti réteggel), és $\mathcal{H}^*(\Theta^*, \cdot) = \mathcal{F}(\Theta^* \setminus \Theta, \cdot) \circ \mathcal{H}(\Theta, \cdot)$ egy \mathcal{H} -nál mélyebb háló, akkor $\mathcal{F}(\Theta^* \setminus \Theta, \cdot) = \text{id}$ esetén \mathcal{H}^* teljesítménye meg kell hogy egyezzen \mathcal{H} teljesítményével.

A gond tehát abban rejlik, hogy a \mathcal{H}^* nehezen tanulja meg a Θ^* paramétereket úgy, hogy \mathcal{F} (közel) identikus legyen. Ezt a problémát próbálja áthidalni a ResNet. Ehhez azt a feltételezést használja, hogy míg az identitásfüggvényt nehéz megtanulni, az $\mathcal{F}(\Theta^* \setminus \Theta, \cdot) \equiv 0$ függvényt könnyebb. Ugyanis ha $\bar{\phi}$ a ReLU aktivációs függvény, $\bar{\phi}(\mathbf{W}\mathbf{x} + \mathbf{b}) = 0$ pontosan akkor teljesül, ha $\mathbf{W}, \mathbf{b} = \mathbf{0}$, ez pedig – legalábbis intuitívan – egy könnyebb feladat.

Ez motiválja a reziduális hálók fő építőelemének, az úgynevezett reziduális egységnek (*residual unit*) a bevezetését.

3.3.1. definíció (reziduális egység). Legyen $\mathbf{f}(\Theta_{\mathbf{f}}, \cdot)$ függvény olyan, hogy $\mathcal{H}^* = \mathcal{H}_1 \circ \mathbf{f} \circ \mathcal{H}_0$ valamilyen \mathcal{H}_0 és \mathcal{H}_1 függvényekre. (Jellemzően \mathbf{f} \mathcal{H}^* néhány rétegének kompozíciója.) Ekkor az

$$\mathbf{f}^*(\Theta_{\mathbf{f}}, \cdot) : \mathbf{x} \mapsto \mathbf{f}(\Theta_{\mathbf{f}}, \mathbf{x}) + \mathbf{x} \quad (3.5)$$

függvényt egy reziduális egységnek nevezem a \mathcal{H}^* hálóban.

A ResNet hálók, az első és utolsó néhány rétegüktől eltekintve, ilyen reziduális egységek kompozíciójaként állnak elő.

A reziduális egységek bevezetése nem növeli a háló komplexitását, hiszen az összeadás \mathbf{x} méretében lineáris, és a tanulható paraméterek számát sem. Megkötés azonban, hogy a fenti definíció csak akkor értelmes, ha $\mathbf{f}(\Theta_{\mathbf{f}}, \cdot)$ megőrzi a bemenet dimenzióját. Amennyiben ez nem teljesül,

$$\mathbf{f}^*(\Theta_{\mathbf{f}}, \cdot) : \mathbf{x} \mapsto \mathbf{f}(\Theta_{\mathbf{f}}, \mathbf{x}) + \mathbf{P}\mathbf{x} \quad (3.6)$$

alakú reziduális egységek lehetségesek, ahol \mathbf{P} lineáris transzformáció. \mathbf{P} együtthatói lehetnek tanulható paraméterek, gyakori azonban ehelyett a $\mathbf{P} : \mathbf{x} \mapsto (\mathbf{x}, \mathbf{0})$ beágyazást venni, ebben az esetben ugyanis továbbra is teljesül, hogy \mathbf{f}^* nem hoz be új paramétereket.

3.3.1. ResNet architektúrák

A következő részben ismertetem a He et al. cikkeiben bemutatott reziduális hálók felépítését.

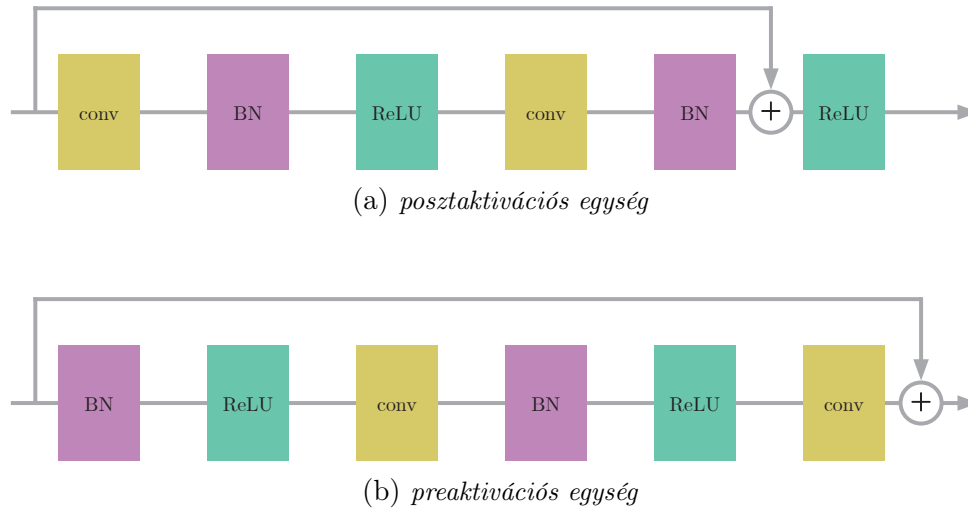
Megjegyzés. Az alábbi hálók szokásos elnevezése ResNet, majd a rétegek száma. Az 56 rétegű ResNet architektúra neve tehát ResNet-56 lesz.

Reziduális egységek

Az első és utolsó két rétegen kívül minden más réteg 3×3 -as konvolúció az összes hálóban, batchnormalizáció (BN) és ReLU aktivációs függvényekkel. A legtöbb háló reziduális egységei két rétegből állnak. Fontos különbség, hogy a reziduális egységben az aktivációs

függvény alkalmazása az összeadás előtt vagy után történik-e, azaz **pre-** vagy **posztaktivációs** egységről beszélünk-e. A két megoldás két, egymással nem ekvivalens hálózathoz vezet, mivel posztaktiváció esetén a reziduális egység kimenetének minden koordinátája nemnegatív, míg preaktiváció esetén ez nem feltétlenül teljesül. Kísérleti tapasztalatok azt mutatják, hogy a preaktivációt használó ResNetek hatékonyabbak.

A 3. ábra bemutatja a kétrétegű reziduális egységeket, 3×3 -as konvolúciókkal.



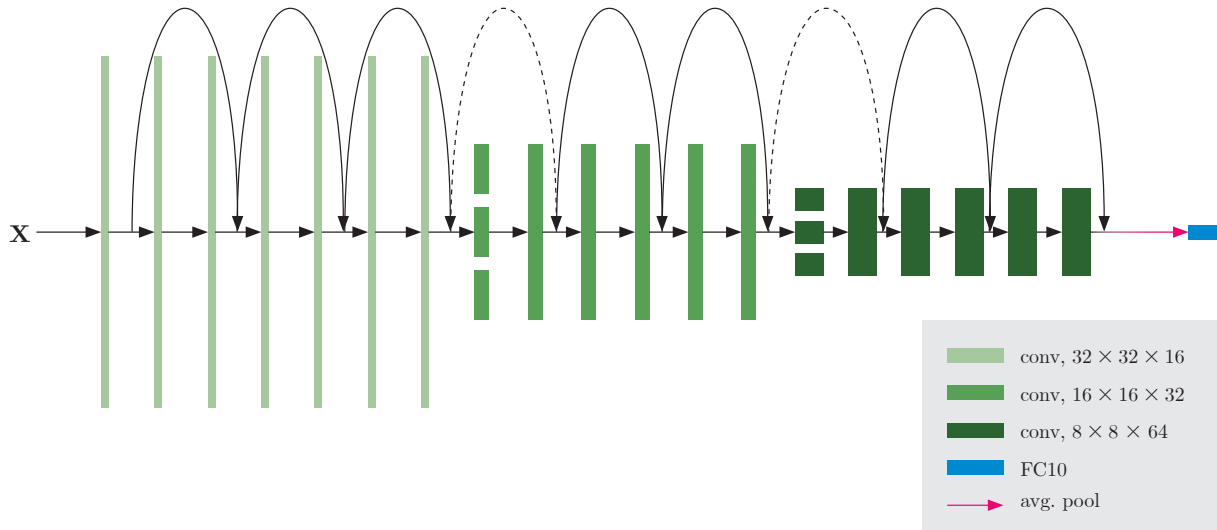
3. ábra. Reziduális egységek felépítése

Mivel a reziduális egység akkor működik a leghatékonyabban, ha a bemenet dimenziója nem változik, a konvolúciók mélysége állandó kell hogy legyen. Ez azonban jelentős számításigénnyel jár mélyebb, ImageNethez épített architektúrák esetén, ahol szükséges akár 256 csatorna mély konvolúciók használata is. Ezért a tanítási idő gyorsítása érdekében He et al. a mélyebb modelleket úgynevezett „palacknyak” (*bottleneck*) szerkezettel építik: a reziduális egység egyetlen 3×3 -as konvolúcióból áll, amelyet két 1×1 -es konvolúció vesz körül. Az első 1×1 -es konvolúció csökkenti a csatornák számát, amelyet az utolsó 1×1 -es konvolúció állít vissza az eredeti méretére. Így a háló komplexitása jelentősen csökken, lehetővé téve ezzel több mint ezer rétegű modellek tanítását is.

A CIFAR-10-en használt architektúrák

A CIFAR-10-re épített sekély ResNetek $6n + 2$ rétegből állnak. Az első réteg egy 3×3 -as konvolúció 16 csatornával, majd ezt követi három darab $2n$ konvolúcióból álló rétegsorozat, amelyek kimenete rendre 32×32 , 16×16 és 8×8 méretű. Minden alkalommal, amikor a térbeli dimenziók feleződnek, a csatornák száma kétszereződik. A dimenzióváltozásokat 2 lépésközi konvolúciók érik el. A háló végén global average pooling és egy sűrű (*fully connected*) réteg található, softmax aktivációval. (A 4. ábra ábrázolja az $n = 3$ esetet.)

He et al. mélyebb hálók esetén $3n + 2$ alakú hálókat használnak, amelyek felépítése megegyezik a sekélyebb hálókéval, de a három darab rétegsorozatban a kétrétegű reziduális egységek helyett háromrétegű bottleneck architektúrák találhatók. A kimenetek térbeli dimenziói változatlanok, de az egyes rétegsorozatban a rétegek mélysége 16, 16, 64, 32, 32, 128, illetve 64, 64, 256.



4. ábra. A ResNet-20 architektúrája. A görbe nyilak összeadást, a szaggatottak beágyazást jelölnek. A szaggatott rétegek 2-es lépésközt használnak

Az ImageNeten használt architektúrák

He et al. [21] cikkében öt, ImageNethez készült ResNet architektúrát mutattak be, 18, 34, 50, 101 és 152 réteggel. Minden háló 224×224 -es bemenettel dolgozik. A reziduális egységek a két sekélyebb háló esetében két konvolúciós rétegből állnak, a mélyebb hálóknál pedig bottleneck struktúrákból. Az 1. táblázatban összefoglalom az egyes hálók felépítését.

Ezek közül a ResNet-152 2015-ben jobb top 5 pontosságot ért el az ImageNeten, mint bármelyik korábbi ILSVRC-nyertes, egy hat ResNet-modellből összetett architektúra pedig megnyerte a 2015-ös versenyt 3,57%-os pontossággal.

Kimenet	Réteg	Darabszám	
		ResNet-18	ResNet-34
$112 \times 112 \times 64$	7×7 conv, $s = 2$	1	1
$56 \times 56 \times 64$	3×3 max pool, $s = 2$	1	1
	3×3 conv	4	6
$28 \times 28 \times 128$	3×3 conv	4	6
$14 \times 14 \times 256$	3×3 conv	4	12
$7 \times 7 \times 512$	3×3 conv	4	6
$1 \times 1 \times 512$	global average pooling	1	1
$1 \times 1 \times 1000$	sűrű réteg	1	1

(a) *sim*a architektúrák

Kimenet	Réteg(ek)	Darabszám		
		ResNet-50	ResNet-101	ResNet-152
$112 \times 112 \times 64$	7×7 conv, $s = 2$	1	1	1
$56 \times 56 \times 64$	3×3 max pool, $s = 2$	1	1	1
$56 \times 56 \times 256$	$\begin{bmatrix} 1 \times 1 \text{ conv, } 64 \\ 3 \times 3 \text{ conv, } 64 \\ 1 \times 1 \text{ conv, } 256 \end{bmatrix}$	3	3	3
$28 \times 28 \times 512$	$\begin{bmatrix} 1 \times 1 \text{ conv, } 128 \\ 3 \times 3 \text{ conv, } 128 \\ 1 \times 1 \text{ conv, } 512 \end{bmatrix}$	4	4	8
$14 \times 14 \times 1024$	$\begin{bmatrix} 1 \times 1 \text{ conv, } 256 \\ 3 \times 3 \text{ conv, } 256 \\ 1 \times 1 \text{ conv, } 1024 \end{bmatrix}$	6	23	36
$7 \times 7 \times 2048$	$\begin{bmatrix} 1 \times 1 \text{ conv, } 512 \\ 3 \times 3 \text{ conv, } 512 \\ 1 \times 1 \text{ conv, } 2048 \end{bmatrix}$	3	3	3
$1 \times 1 \times 2048$	global average pooling	1	1	1
$1 \times 1 \times 1000$	sűrű réteg	1	1	1

(b) „bottleneck” architektúrák

1. táblázat. Az ImageNeten használt ResNetek felépítése. „ $s = 2$ ” kettes lépésközt jelöl

4. fejezet

Az LM-ResNet

Y. Lu, A. Zhong, Q. Li és B. Dong [23] cikkükben egy módosítást mutatják be He et al. ResNetjének, amelyet LM-ResNetnek neveznek. Ebben a fejezetben bemutatom az általuk használt architektúrát, illetve ismertetem az eredményeiket.

4.1. ResNet mint differenciálegyenlet-közelítés

Az LM-ResNet architektúra motivációja He et al. ResNetje és az explicit Euler-módszer (1.6) összefüggése. Tekintsünk ugyanis egy \mathcal{H} preaktivációs reziduális hálót, valamint tekintsük az

$$\mathbf{x}_{i+1} = \mathbf{f}_i(\Theta_i, \mathbf{x}_i) + \mathbf{x}_i \quad (4.1)$$

differenciaegyenletet. Ekkor ha \mathcal{H} m reziduális egységből áll, akkor $\mathbf{x}_0 = \mathbf{f}_{-1}(\mathbf{x})$ esetén $\mathcal{H}(\mathbf{x}) = \mathbf{f}_{m+1}(\mathbf{x}_m)$, ahol \mathbf{f}_{-1} az első reziduális egység előtt lévő rétegek kompozíciója, \mathbf{f}_{m+1} pedig a háló végén található pooling, sűrű réteg és softmax aktiváció kompozíciója. (Most elhanyagolva az esetleges (3.6) szerinti \mathbf{P} projekciókat.)

A fontos észrevétel az, hogy eltekintve attól a pár rétegtől, ahol \mathbf{x}_i dimenziója megváltozik, az \mathbf{f}_i függvények megegyeznek, mivel a rétegek közti különbségeket csak a Θ_i paraméterek szolgáltatják. Ezért a (4.1) egyenletben most \mathbf{f}_i helyett $\Delta t \mathbf{f}_i$ -t írva, \mathcal{H} rejtett rétegei felfoghatók mint az

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\Theta(t), \mathbf{x}(t)) \quad (4.2)$$

differenciálegyenlet explicit Euler-módszer adta közelítései.

Megjegyzés. Δt beírása a (4.1) egyenletbe triviális, hiszen az egyenletben már explicit szerepel a $\Delta t = 1$ szorzótényező. Azonban $\Delta t > 0$ esetén $\Delta t \mathbf{f}(\Theta_i, \cdot) = \mathbf{f}(\Delta t \Theta_i, \cdot)$, mivel \mathbf{f} nemlineáris komponensei a batchnorm és ReLU függvények, azonban mindkettő felcserélhető a pozitív számmal való szorzással. Így feltehető akár, hogy $t \in [0, 1]$ és $\Delta t = \frac{1}{m}$. Ez összhangban áll azzal, hogy mély ResNetekben $\mathbf{f}(\Theta_i, \cdot)$ $\mathbf{0}$ -hoz közel van. [24]

Megjegyzés. Bár a fenti felírás egy jó intuíció, nehéz lenne precízzé tenni, mivel egy folytonos dinamikai rendszer nem tudja kezelni a dimenzióváltást, amely része a legtöbb ResNet architektúrának. Azonban tipikusan a ResNetek csupán három különböző dimenziójú részből tevődnek össze (eltekintve az első és utolsó néhány rétegtől, amely nem része a reziduális architektúrának), így \mathcal{H} felfogható mint három különböző, állandó szélességű ResNet kompozíciója, amelyek mindegyike egy (4.2) alakú egyenletet kielégítő függvényt approximál.

Belátható ekkor, hogy az állandó szélességű ResNetek a megfelelő feltételek mellett $m \rightarrow \infty$ esetén tartani fognak pontonként a (4.2) egyenlet \mathbf{x} megoldásához. [25]

Lu et al. fő ötlete az, hogy a (4.1) egyenletben használt explicit Euler-módszer helyett a (4.2) differenciálegyenlet közelítésére egy (1.11) lineáris többlépéses módszert alkalmazzanak. Speciálisan az LM-ResNet architektúrák az alap ResNet reziduális egységeit

$$\mathbf{x}_{i+1} = (1 - \vartheta_i)\mathbf{x}_i + \vartheta_i\mathbf{x}_{i-1} + \mathbf{f}(\Theta_i, \mathbf{x}_i) \quad (4.3)$$

többlépéses egységekkel helyettesítették, ahol ϑ_i az i . reziduális egységhez tartozó tanulható paraméter.

Megjegyzés. A ϑ_i értékek miatt egy LM-ResNetnek több tanulható paramétere van, mint a vele azonos rétegszámú ResNetnek. Ez azonban szinte elhanyagolható különbség, mivel a ResNet-20-nak is több mint negyedmillió paramétere van, az LM-ResNet-20 modell pedig mindössze kilenc új paramétert hoz. Ezért pusztán a paraméterek száma nem indokolja jelentős különbséget a két modell teljesítménye között.

4.2. Implementáció

Modellek

Minden ResNet (és más, ResNet-szerű architektúrák) módosítható LM-architektúrává. Lu et al. a CIFAR-10-en 20, 32, 44 és 56 rétegű LM-ResNeteket tesztelnek, amelyek a 28. oldalon ismertetett felépítésűek. CIFAR-100-on 110 és 164 réteg mély LM-ResNeteket teszteltek, bottleneck architektúrákkal, ImageNeten pedig az 1. táblázatban bemutatott 50 és 101 rétegű ResNetek változatait. Minden háló preaktivációs megvalósítást alkalmazott.

Adathalmazok

Lu et al. három adathalmazon végeztek méréseket: CIFAR-10, CIFAR-100 és ImageNet. A CIFAR adathalmazok tanító adatpontjain Lee et al. [26] előfeldolgozását és augmentációját használják, ugyanúgy, ahogyan az eredeti ResNet cikkek. A képpixelek intenzitását $[0, 1]$ -be normálták, majd globális kontrasztnormalizációt (3.2) alkalmaztak. A tanító adatpontok minden oldalát kipárnázták négy-négy (0, 0, 0) (fekete) pixellel, majd ebből vagy a vízszintes tükörképéből véletlenszerűen kivágnak egy 32×32 -es darabot.

Az ImageNeten Krizhevsky et al. [27] alapján előfeldolgozásként a tanító adathalmaz pixeljeiből kivonták az adathalmaz pixelintenzitásainak átlagát, így a pixelértékeket $[-1, 1]$ -beli, 0 átlagú értékek lesznek. Tanítás során a képeket átméretezték, hogy a rövidebb oldaluk eloszlása $[256, 480]$ -on diszkrét egyenletes legyen, majd az így kapott képből véletlenszerűen kivágtak egy 224×224 -es darabot.

Inicializáció és hiperparaméterek

He et al. [28] alapján egy d bemeneti kapcsolattal rendelkező neuron súlyait $\mathcal{N}(0, \frac{2}{d})$ eloszlásból inicializálták, a biasokat 0-val, a (4.3) többlépéses reziduális egység ϑ_i tanulható paramétereit pedig $\mathcal{U}[-\frac{1}{10}, 0]$ eloszlásból.

A tanításhoz SGD optimalizációt használtak, a CIFAR-on 128-as, az ImageNeten 256-os batchmérettel, $\frac{9}{10}$ momentummal és 10^{-4} súlyelhalással (*weight decay*). A tanítást 10^{-1} tanulási rátával kezdték, majd ezt tizedelték CIFAR-10 esetén a 80. és 120., CIFAR-100 esetén a 150. és 225. epochnál. CIFAR-10-en 160, CIFAR-100-on 300 epochon keresztül tanítottak.

Megjegyzés. Az implementációs részletek egytől egyig megegyeznek a He et al. [21, 22] által használt implementációval, kivéve hogy Lu et al. mérései végig preaktivációs reziduális egységeket használtak, míg He et al. [21] cikkben közölt eredményei posztaktivációs modellekre vonatkoznak. Mivel a paraméterek száma között is csak elhanyagolható különbség van, ezért a mért eredmények közti különbség szinte kizárólag a (4.3) többlépcsős reziduális egységek alkalmazásának tudható be.

4.3. Eredmények

Az alábbiakban ismertetem Lu et al. különböző LM-ResNetjeinek teljesítményét a CIFAR-10, CIFAR-100 és ImageNet adathalmazokon, valamint a hasonló architektúrájú ResNetek teljesítményét. A posztaktivációs ResNetek eredményei a [21], a preaktivációs ResNeteké a [22] cikkekben jelentek meg, kivéve a ResNet-110 CIFAR-100-on mért teljesítményét, amelyet Huang et al. közöltek a [29] cikkben, de a használt architektúra megegyezik a korábban ismertettekkel.

Minden modell esetében a „hiba” a tesztadathalmazon mért hibára vonatkozik.

Adathalmaz	Modell	Reziduális egység	Hiba
CIFAR-10	ResNet-20	posztaktivációs	8,75%
CIFAR-10	ResNet-32	posztaktivációs	7,51%
CIFAR-10	ResNet-44	posztaktivációs	7,17%
CIFAR-10	ResNet-56	posztaktivációs	6,97%
CIFAR-10	ResNet-110	preaktivációs	6,37%
CIFAR-10	LM-ResNet-20	preaktivációs	8,33%
CIFAR-10	LM-ResNet-32	preaktivációs	7,18%
CIFAR-10	LM-ResNet-44	preaktivációs	6,66%
CIFAR-10	LM-ResNet-56	preaktivációs	6,31%
CIFAR-100	ResNet-110	posztaktivációs	27,76%
CIFAR-100	ResNet-164	preaktivációs	24,33%
CIFAR-100	ResNet-1001	preaktivációs	22,71%
CIFAR-100	LM-ResNet-110	preaktivációs	25,87%
CIFAR-100	LM-ResNet-164	preaktivációs	22,90%

2. táblázat. ResNet és LM-ResNet architektúrák hibája a CIFAR adathalmazokon

A 2. táblázatban összefoglaltam a CIFAR adathalmazokon mért eredményeket. Látható, hogy minden LM-ResNet jelentősen jobban teljesített, mint az azonos felépítésű sima

reziduális háló. CIFAR-10 esetében ezt a különbséget részben okozhatja az, hogy az LM-ResNetek preaktivációt alkalmaznak posztaktiváció helyett, azonban nem ez az egyetlen oka a javulásnak, hiszen az 56 rétegű LM-ResNet jobban teljesít, mint a 110 rétegű, preaktivációt alkalmazó ResNet.

Hasonló tendencia figyelhető meg a CIFAR-100 adathalmazon: az LM architektúrák közelítőleg elérik egy jóval mélyebb ResNet hibáját. A 164 réteg mély LM-ResNet képes majdnem olyan jó eredményt elérni, mint a ResNet-1001.

Adathalmaz	Modell	Reziduális egység	Top 1 hiba	Top 5 hiba
ImageNet	ResNet-50	posztaktivációs	24,7%	7,8%
ImageNet	ResNet-101	posztaktivációs	23,6%	7,1%
ImageNet	ResNet-152	posztaktivációs	23,0%	6,7%
ImageNet	ResNet-152	preaktivációs	21,1%	5,5%
ImageNet	LM-ResNet-50	preaktivációs	23,8%	7,0%
ImageNet	LM-ResNet-101	preaktivációs	22,6%	6,4%

3. táblázat. *ResNet és LM-ResNet architektúrák top 1 és top 5 hibája ImageNeten*

A 3. táblázatban láthatók az ImageNeten mért eredmények. Itt is látható javulás LM-architektúrák alkalmazása esetén, azonban ez kevésbé jelentős, és nem egyértelmű, hogy ez mennyiben tudható be a preaktivációs modellek alkalmazásának, ugyanis az ImageNeten He et al. nem publikáltak méréseket 152-nél sekélyebb preaktivációs modellel, Lu et al. pedig csupán az 50 és 101 réteg mély LM-ResNeteket tesztelték.

4.4. A kísérletek reprodukálása

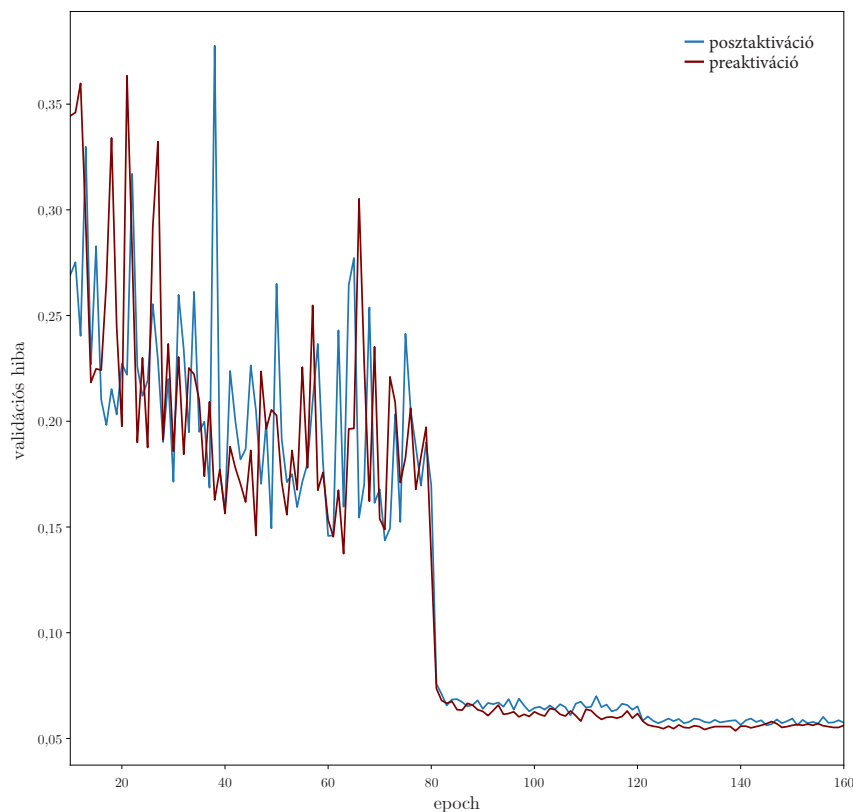
A következőkben bemutatom a saját méréseimet a különböző mélységű ResNetek és LM-ResNetek teljesítményéről a CIFAR-10 adathalmazon. A modellek létrehozásához, valamint a kísérletek lefuttatásához a TensorFlow programkönyvtárat [30] és a Keras API-t [31] használtam.

A következő részekben ismertetem a tapasztalataimat. Az eredményeket részletesen összefoglalom a 4. táblázatban.

4.4.1. ResNet eredmények

A 20, 32, 44, 56, 110 és 164 réteg mély ResNetek teljesítményét mértem, mind pre-, mind posztaktivációs reziduális egységekkel. Minden mérést ötször futtattam le, különböző paraméterinicializációkkal. A mérések megerősítik, hogy a preaktivációs egységek erősebb hálókhoz vezetnek. Ez alól kivétel a ResNet-20, ahol posztaktiváció esetén az öt futtatás átlagosan kisebb hibát eredményezett, bár a legjobb, 8,12%-os hibát preaktivációs egységekkel mértem.

Posztaktivációs egységek esetén az átlagos mérések rendre rosszabbak lettek He et al. eredményeinél, azonban az öt futtatás közül a legjobb, a ResNet-110 kivételével minden



5. ábra. A ResNet-164 validációs hibája pre-, illetve posztaktivációs reziduális egységek esetén¹

esetben jobb eredményt hozott, mint az eredeti mérések. Mindkét jelenségnek lehet az a magyarázata, hogy a ResNet-110 kivételével He et al. eredményei egyetlen mérésből származnak.

Preaktivációs esetben He et al. nem közöltek eredményeket a sekélyebb hálókra. A ResNet-110 és ResNet-164 preaktivációs változata esetén ugyanaz tapasztalható, mint a posztaktivációs hálónál: He et al. eredményei az általam mért átlagos és legjobb eredmények közé esnek.

4.4.2. LM-ResNet eredmények

A Lu et al. cikkében közzétett (4.3) képlet szerinti architektúrának két hiányossága van. Egyrészt, azzal analóg módon, hogy egy lineáris kétlépéses módszer is csak akkor alkalmazható, ha az \mathbf{x}_1 első közelítést valamilyen más módszerrel találjuk meg, a (4.3) képlet \mathbf{x}_{i+1} -re is csak $i \geq 1$ esetén értelmes, így a ResNet első reziduális egységét nem lehet az LM-architektúra szerint módosítani.

Itt két megoldást próbáltam ki. Az első (továbbiakban A) módszer során az

$$\mathbf{x}_1 = \mathbf{f}_1(\Theta_1, \mathbf{x}_0) + \mathbf{x}_0 \quad (4.4)$$

szabállyal határoztam meg \mathbf{x}_1 -et, ahol \mathbf{x}_0 az első reziduális egység bemenete, \mathbf{f}_1 pedig az első két konvolúciós réteg kompozíciója. A második (továbbiakban B) módszerben pedig

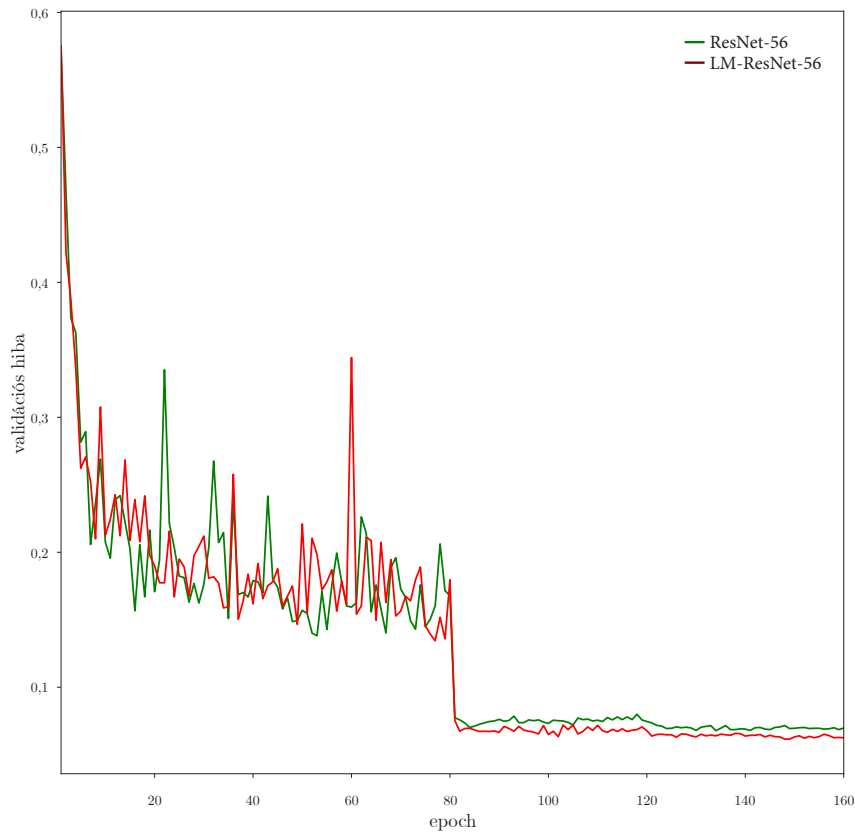
¹A vízszintes tengely a 10. epochnál kezdődik az ábra méretének csökkentése érdekében, ugyanis ez előtt a hiba nagyon magas volt.

a

$$\mathbf{x}_1 = \mathbf{f}_1(\Theta_1, \mathbf{x}_0) + \mathbf{P}(\Theta_P)\mathbf{x}_0 \quad (4.5)$$

szabályt alkalmaztam, ahol $\mathbf{P}(\Theta_P)$ 1×1 -es konvolúció tanulható paraméterekkel. Ez a módszer ugyan további új paramétereket vezet be, de egy 16 csatorna mély rétegek közötti 1×1 -es konvolúciónak pusztán 272 paramétere van, amely még a legsekélyebb háló, a ResNet-20 esetén is pusztán az ezrede az összes paraméternek.

Megjegyzés. A fentiek közül (4.4) egy sima reziduális egységet ír le. A (4.5) projekciót használó egységet He et al. is felvetették [21], és ők is azt tapasztalták, hogy az ilyen típusú egységek alkalmazása kis mértékben ugyan, de javít a háló teljesítményén, feltehetően a tanulható paraméterek megnövekedett száma miatt.



6. ábra. A ResNet-56 (preaktivációval) és az LM-ResNet-56 (B változat) validációs hibája

Lu et al. arra sem tértek ki, hogy a (4.3) szabály hogyan módosul, ha \mathbf{x}_{i+1} és \mathbf{x}_{i-1} dimenziója nem egyezik meg. Ilyen esetekben mindig a $\mathbf{P} : \mathbf{x} \mapsto (\mathbf{x}, \mathbf{0})$ projekciót használtam, ugyanis ha itt is 1×1 -es konvolúciókat alkalmaztam volna, az már jelentősen megnövelte volna a paraméterek számát.

Az LM-ResNet 20, 32, 44, 56 és 110 réteg mély változatán végeztem méréseket. Az A változatú hálóknak mindegyikén négyszer, a B változatú hálókon ötször futtattam le a méréseket.

A B változatot használó modellek valamivel jobban teljesítettek az A változatoknál, azonban ezek is átlagosan mintegy 3%-kal nagyobb hibával teljesítnek, mint Lu et al. közölt eredményei. Ennek oka többféle lehet. Egyrészt, mivel Lu et al. nem közölték az architektúra részletes leírását és nem biztosítottak kódot a modelljeikhez, lehetséges, hogy bizonyos implementációs részletekben más döntéseket hoztam, mint ők. Másrészt Lu et

al. a modelljeiket a CIFAR-10 50 000-es tanító adathalmazán tanították, míg én csak 45 000 képen tanítottam és 5000 képet validációra használtam, összhangban He et al. módszerével, amelyet a ResNeten végzett mérések során alkalmaztak.

Modell	Reziduális egység	Hiba		
		Átlag	Legjobb	Eredeti
ResNet-20	posztaktivációs	8,34%	8,19%	8,75%
ResNet-32	posztaktivációs	7,67%	7,29%	7,51%
ResNet-44	posztaktivációs	7,94%	7,16%	7,17%
ResNet-56	posztaktivációs	7,60%	6,89%	6,97%
ResNet-110	posztaktivációs	7,19%	6,79%	6,61%
ResNet-164	posztaktivációs	6,28%	5,90%	5,93%
ResNet-20	preaktivációs	8,58%	8,12%	–
ResNet-32	preaktivációs	7,64%	7,51%	–
ResNet-44	preaktivációs	7,09%	6,98%	–
ResNet-56	preaktivációs	6,76%	6,61%	–
ResNet-110	preaktivációs	6,28%	6,06%	6,37%
ResNet-164	preaktivációs	5,52%	5,18%	5,46%
LM-ResNet-20 A	preaktivációs	8,62%	8,37%	8,33%
LM-ResNet-20 B	preaktivációs	8,52%	8,45%	
LM-ResNet-32 A	preaktivációs	7,77%	7,60%	7,18%
LM-ResNet-32 B	preaktivációs	7,63%	7,28%	
LM-ResNet-44 A	preaktivációs	7,10%	6,89%	6,66%
LM-ResNet-44 B	preaktivációs	7,02%	6,84%	
LM-ResNet-56 A	preaktivációs	6,91%	6,65%	6,31%
LM-ResNet-56 B	preaktivációs	6,58%	6,28%	
LM-ResNet-110 A	preaktivációs	6,07%	5,80%	–
LM-ResNet-110 B	preaktivációs	6,17%	5,98%	

4. táblázat. Az általam mért átlagos és legjobb teljesítmények, valamint az eredeti (He et al., illetve Lu et al. által publikált) eredmények a CIFAR-10 adathalmazon

A valamivel rosszabb eredmények ellenére az LM-ResNet architektúrák rendre jobb átlagos teljesítményt nyújtottak, mint az azonos mélységű ResNetek, sőt az LM-ResNet öt mérés alapján számolt átlagos hibája a mélyebb hálók esetén jobb volt, mint a velük megegyező mélységű sima ResNetek legjobb mért hibája.

Ezek alapján kijelenthető, hogy az LM-architektúra javítja a reziduális hálók teljesítményét. Ez a javulás nem tudható be pusztán annak, hogy Lu et al. preaktivációs hálókat hasonlított össze azonos mélységű, de posztaktivációt használó modellekkel: a méréseim azt mutatják, hogy az LM-ResNetek a megfelelő mélységű preaktivációs ResNeteknél is

jobb teljesítményt nyújtanak. Szintén valószínűtlen, hogy a megnövekedett paraméterszám okozza a különbséget, ugyanis mindegyik hálónak negyed- és kétmillió közti paramétere van, az LM architektúrával pedig kevesebb mint háromszázzal változik ez a szám.

Ennek következtében a Lu et al. és általam is tapasztalt javulás legalábbis részben az új, lineáris többlépéses módszeren alapuló architektúrának köszönhető. Ez az eredmény is jó példája annak, hogy érdemes vizsgálni a reziduális hálók és a differenciálegyenletek között megjelenő természetes összefüggéseket. Ez, mint láttuk, egyes esetekben jobb teljesítményű hálóhoz vezet. A gyakorlati hasznon kívül azonban az sem kizárt, hogy ez az új nézőpont a segítségünkre lesz majd abban, hogy jobban megértsük a mesterséges neuronhálókat, és azon belül is a ResNetek és a hozzájuk hasonló architektúrák működését.

Irodalomjegyzék

- [1] Faragó István – Horváth Róbert: *Numerikus módszerek*. Elektronikus kiadás, Typotex Kiadó, 2013. (230–235., 256–264. o.)
- [2] Ernst Hairer – Syvert P. Nørsett – Gerhard Wanner: *Solving Ordinary Differential Equations I: Nonstiff Problems*. Elektronikus kiadás, Springer, 2008. (368–396. o.)
- [3] Simon L. Péter – Tóth János: *Differenciálegyenletek: Bevezetés az elméletbe és az alkalmazásokba*. Elektronikus kiadás, Typotex Kiadó, 2005. (44–45. o.)
- [4] Stoyan Gisbert – Takó Galina: *Numerikus módszerek 2*. Elektronikus kiadás, Typotex Kiadó, 2012. (15., 81–83., 109–110. o.)
- [5] Charles F. Van Loan: The ubiquitous Kronecker product. *Journal of Computational and Applied Mathematics* **123** (1–2), 2000. november (85–100. o.)
- [6] Charu C. Aggarwal: *Neural Networks and Deep Learning: A Textbook*. Elektronikus kiadás, Springer, 2018. (loc. 604–2208., 3901–5924., 10 463–11 657.)
- [7] Álmos Attila – Győri Sándor – Horváth Gábor – Várkonyiné Kóczy Annamária: *Genetikus algoritmusok*. Elektronikus kiadás, Typotex Kiadó, 2003. (139–151. o.)
- [8] Allan Pinkus: Approximation theory of the MLP model in neural networks. *Acta Numerica* **8**, 1999. január (143–195. o.)
- [9] Robert M. Burton és Herold G. Dehling: Universal approximation in p -mean by neural networks. *Neural Networks* **11**, 1998. június (661–667. o.)
- [10] Patrick Kidger és Terry Lyons: Universal approximation with deep narrow networks. *Proceedings of Thirty Third Conference on Learning Theory*, 2020. július 9–12. PMLR (2306–2327. o.)
- [11] Sejun Park, Chulhee Yun, Jaeho Lee és Jinwoo Shin: Minimum width for universal approximation. *International Conference on Learning Representations*, 2021. május 3–7. <https://openreview.net/forum?id=0-XJwyoIF-k>. Letöltés dátuma: 2021. március 19.
- [12] Zeyuan Allen-Zhu, Yuanzhi Li és Zhao Song: A convergence theory for deep learning via over-parameterization. *arXiv preprint*, 2019. június 17. arXiv:1811.03962v5 [cs.LG]
- [13] Sandro Skansi: *Introduction to Deep Learning: From Logical Calculus to Artificial Intelligence*. Elektronikus kiadás, Springer, 2018. (109–111., 118–119. o.)
- [14] Ilya Loshchilov és Frank Hutter: Decoupled weight decay regularization. *arXiv preprint*, 2019. január 4. arXiv:1711.05101v3 [cs.LG]

-
- [15] Alex Krizhevsky: Learning Multiple Layers of Features from Tiny Images, 2009. <https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf>. Letöltés dátuma: 2021. március 30.
- [16] Jia Deng, Wei Dong, R. Socher, L. Li, Kai Li és Li Fei-Fei: ImageNet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009. június 20–25. IEEE (248–255. o.)
- [17] Olga Russakovsky, Jia Deng, Hao Su et al.: ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision* **115**, 2015. április (221–252. o.)
- [18] Tanya Dayanand: Data Preprocessing and Network Building in CNN. *Towards Data Science*. <https://towardsdatascience.com/data-preprocessing-and-network-building-in-cnn-15624ef3a28b>. Letöltés dátuma: 2021. április 3.
- [19] Ian Goodfellow – Yoshua Bengio – Aaron Courville: *Deep Learning*. Elektronikus kiadás, MIT Press, 2016. (448–453. o.)
- [20] Min Lin, Qiang Chen és Shuicheng Yan: Network In Network. *arXiv preprint*, 2014. március 4. arXiv:1312.4400v3 [cs.NE]
- [21] Kaiming He, Xiangyu Zhang, Shaoqing Ren és Jian Sun: Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition*, 2016. június 27–30. IEEE (770–778. o.)
- [22] Kaiming He, Xiangyu Zhang, Shaoqing Ren és Jian Sun: Identity mappings in deep residual networks. *Computer Vision – ECCV 2016*, 2016. október 8–16. Springer (630–645. o.)
- [23] Yiping Lu, Aoxiao Zhong, Quanzheng Li és Bin Dong: Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. *Proceedings of the 35th International Conference on Machine Learning*, 2018. június 10–15. PMLR (3276–3285. o.)
- [24] Soham De és Samuel L. Smith: Batch normalization biases residual blocks towards the identity function in deep networks. *Advances in Neural Information Processing Systems* **33**. Curran Associates, 2020. (19964–19975. o.)
- [25] Matthew Thorpe és Yves van Gennip: Deep limits of residual neural networks. *arXiv preprint*, 2020. október 13. arXiv:1810.11741v3 [math.CA]
- [26] Chen-Yu Lee, S. Xie, P. W. Gallagher, Z. Zhang és Z. Tu: Deeply-supervised nets. *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, 2015. május 9–12. PMLR (562–570. o.)
- [27] Alex Krizhevsky, Ilya Sutskever és Geoffrey E. Hinton: ImageNet classification with deep convolutional neural networks. *Communications of the ACM* **60**, 2017. június (84–90. o.)
- [28] Kaiming He, Xiangyu Zhang, Shaoqing Ren és Jian Sun: Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. *2015 IEEE International Conference on Computer Vision (CVPR)*, 2015. december 7–13. IEEE (1026–1034. o.)
- [29] Gao Huang, Yu Sun, Z. Liu, D. Sedra és K. Q. Weinberger: Deep networks with stochastic depth. *Computer Vision – ECCV 2016*, 2016. október 8–16. Springer (646–661. o.)
- [30] Martín Abadi, Ashish Agarwal, Paul Barham et al.: TensorFlow: Large-scale machine learning on heterogeneous distributed systems, 2015. Szoftver elérhető a <https://www.tensorflow.org> címen
- [31] François Chollet: Keras, 2015. <https://keras.io>