EÖTVÖS LORÁND TUDOMÁNYEGYETEM

TERMÉSZETTUDOMÁNYI KAR

GÉPI TANULÁS ALKALMAZÁSA MOLEKULÁRIS BIOLÓGIAI FELADATOKRA

SZAKDOLGOZAT

Szökrön Dorottya

Matematika BSc

Matematikai elemző szakirány

Témavezető: Lukács András

Számítógéptudományi Tanszék

Budapest

2021

Köszönetnyilvánítás

Szeretném köszönetemet kifejezni Lukács András témavezetőmnek és tanáromnak, aki oktatói tevékenysége által megismertette és megszerettette velem a matematika ezen területét. Valamint hálával tartozom neki, hogy szakdolgozatom elkészítéséhez komoly szakmai segítséget biztosított, és motivált ezen témakör mélyebb tanulmányozására.

Köszönöm családomnak a kitartó támogatást és a fáradhatatlan biztatást.

Tartalomjegyzék

1. fejezet

Bevezetés

Szakdolgozatom témájának kiválasztásában a mesterséges intelligencia témaköre iránti lelkesedésem motivált. A matematikatudomány területe kiválló alapot biztosít e témakör megismerésére, például a gépi tanulás különböző módszerein keresztül. Ezen eljárások rendkívűl kiterjedt módon hasznosíthatóak. Számos olyan alkalmazásuk létezik, amely túlmutat a matematika világán, így kapcsolva össze azt olyan eltérő tudományterületekkel, mint a biológia, a zenetudomány vagy a művészetek. Számomra a gépi tanulás által megteremthető tudományköziség is imponáló volt. Ebben a dolgozatban sokféle módszer és feladat közül a mély tanulás bizonyos szegmensét mutatom be egy biológiai feladaton keresztül, amely a fehérjék osztályozására irányul.

Dolgozatomban először bemutatásra kerülnek a téma megértéséhez szükséges alapfogalmak, majd az alapokon túlmutató fogalmak részletes kifejtése következik. Dolgozatom második felében a feladat megoldásával, és a megoldások érétkelésével foglalkoztam.

2. fejezet

Alapfogalmak

2.1. Biológiai alapfogalmak

2.1.1. Fehérjék

Ebben a fejezetben az [5] forrás segítségével fogom bemutatni a fehérjékhez kapcsolódó alapfogalmakat.

A fehérjék az élő szervezet legelterjedtebb szerves molekulái közé tartoznak. Más makromolekulaosztályokhoz képest szerkezetük és működésük sokkal változatosabb. Egyetlen sejt több ezer fehérjét tartalmazhat, amelyek közül mindegyik egyedi funkcióval rendelkezik. Mindezen szerkezeti és működési változatosságuk ellenére mindegyik fehérje egy aminosavlánc, amely ugyanazon 20 féle aminosavból épülnek fel.

Néhány példa a fehérjék szervezetben betöltött különböző szerepei közül.

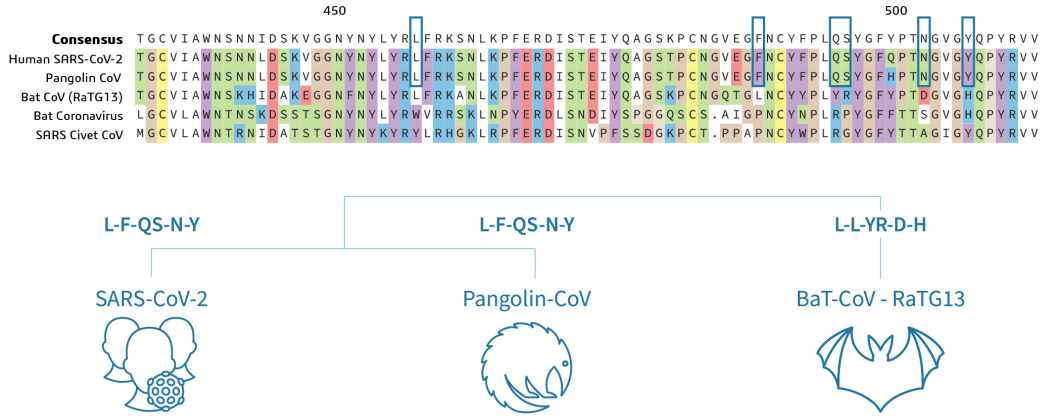
Az enzimek biokémiai reakciókban katalizátorként működnek, azaz felgyorsítják a reakciókat. Szubsztrátoknak nevezik azokat a molekulákat, amelyekkel egy adott reakció elindul, ezeket ismerik fel az enzimek. Az enzimek különböző típusúak lehetnek, és ez alapján különböző reakciókban tudnak részt venni, például lebonthatják, összekapcsolhatják vagy átrendezhetik a felismert szubsztrátokat.

Az endokrin sejtek (belső elválasztású mirigyek) bocsájtják ki a hormonokat, tehát azok mondhatni nagy távolságú kémiai jelek. Különböző fiziológiai folyamatok irányításáért felelősek, például növekedés, anyagcsere, szaporodás. Egyes hormonok szteroid, egyesek pedig fehérje alapúak. Utóbbiakat szokás peptid hormonoknak nevezni.

A fehérjék szerkezeti szempontból sokféle alakban és méretben fordulnak elő. Léteznek gömb alakúak, ilyen például a vérben oxigént szállító hemoglobin. Ugyanakkor léteznek hosszú, vékony szálakat, azaz rostokat alkotók is, például a bőrünkben található kollagén. Azért fontos a geometriai tulajdonságokat is figyelembe venni, mert ezek befolyásolják a fehérje funkcióját is. Sőt ha valamilyen hatás miatt megváltozik a fehérje alakja, akkor korábbi funkcióját vagy teljes funkcionalitását is elveszítheti.

A fentebb említett aminosavak a fehérjéket alkotó monomerek. A monomerek olyan molekulák melyek képesek összekapcsolódni és így alkotni egy láncot, amit polimer láncnak nevezünk, fehérjék esetében polipeptidláncnak. Ezeken kívül létezik néhány „nem kanonikus” aminosav, amelyek ritkán találhatók meg a fehérjékben.

A fehérjéket meghatározzák az őket alkotó polipeptidláncok, azon belül pedig nagyon lényeges az aminosavak kapcsolódásának sorrendje. Ezen kapcsolódási sorrendet nevezzük aminosav szekvenciának, és ez adja a fehérjék elsődleges szerkezetét.



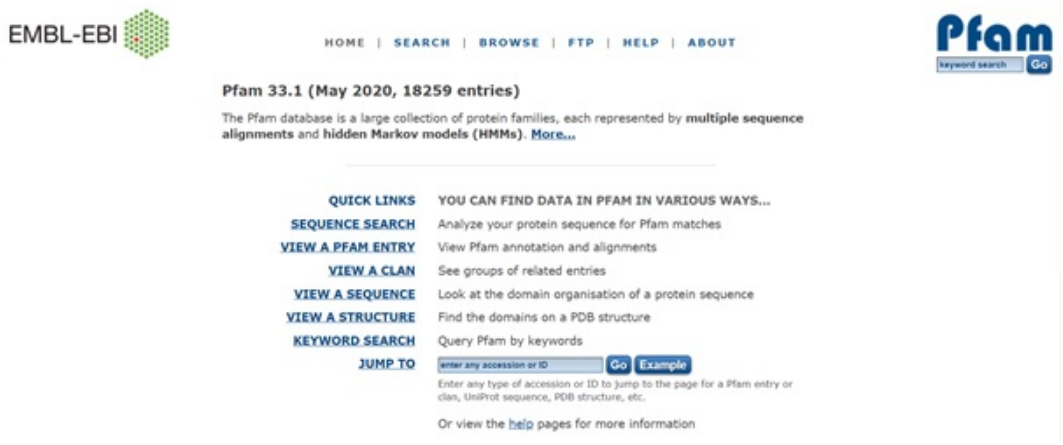
2.1. ábra. Koronavírus variánsok összehaonlítása. [11]

2.1.2. Fehérjeadatbázisok

Ebben a fejezetben a [12], [13] források segítségével fogom bemutatni a fehérjeadatbázisokat.

PFAM

A fehérjeszekvenciák osztályozásához a legszélesebb körben használt, folyamatosan frissülő adatbázis a PFAM (Protein Families Database).



2.2. ábra. PFAM adatbázis. [12]

Elsődleges célja a fehérjecsaládok teljes és pontos osztályozásának biztosítása. Eredetileg azért hozták létre, hogy legyen egy automatizált módszer a már ismert fehérjecsaládokról szerzett információk kezelésére, amely megkönnyíti a genomok jellemzését. Így jött létre a fehérjék PFAM osztályozása, amit a biológusok mára elfogadottnak tekintenek és különböző területeken hasznosítják. Például a kísérleti biológusok sajátos tulajdonságokkal rendelkező fehérjéket kutatnak, a szerkezeti biológia területén az új szerkezetek meghatározása a cél, a bioinformatikusok a fehérjeszekvenciák rendszerezésével foglalkoznak és az evolúcióbiológia tudománya foglalkozik a fehérjék eredetével.

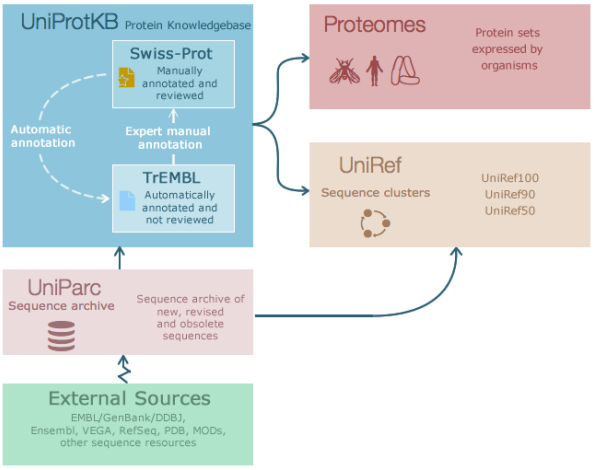
Lényegében a PFAM a fehérjedomén családok gyűjteménye, ahol minden családot egy Multiple Sequence Alignment (MSA) és egy rejtett Markov modell (HMM) reprezentál. Az MSA hasonló hosszúságú biológiai szekvenciák (fehérje vagy nukleinsav) összehangolása, melynek kimenete alapján homológiákat vagy a szekvenciák közötti evolúciós összefüggéseket lehet felfedezni.

A PFAM ingyenesen elérhető bárki számára, alkalmas böngészésre és a szükséges adatok

letöltésére is. A legújabb verzió, a PFAM 33.1 2020. májusában jelent meg és 18 259 családot tartalmaz. Elérhetősége: http://pfam.xfam.org/.

UniProt

Szintén átfogó és szabadon elérhető adatbázis az UniProt (Universal Protein Resource), mely fehérjeszekvenciákat és azok biológiai működésével kapcsolatos információkat tartalmaz. Az adatbázis alkotó részei az UniProt Knowledgebase, (UniProtKB), az UniProt Reference Clusters (UniRef), és az UniProt Archive (UniParc).



2.3. ábra. UniProt adatbázis.[13]

Az UniProtKB két részből áll, az UniProtKB/Swiss-Prot és az UniProtKB/TrEMBL. Az előbbiben található rekordok manuálisan létrehozott jellemzéseket tartalmaznak, melyek szakirodalomból származnak, vagy olyan számítógép által kapott eredményeket, amleyek biológusok által ellenőrzésre kerültek. Ezzel szemben az UniProtKB/TrEMBL számítógépes elemzés során automatikusan kapott adatokat tartalmaz.

Elérhetősége: https://www.uniprot.org/.

2.2. Matematikai alapfogalmak

Ebben a fejezetben a [3], [4] források segítségével fogom bemutatni a témához kapcsolódó matematikai alapfogalmakat.

2.2.1. Neurális hálózatok felépítése és működése

A neurális hálózatok a központi idegrendszer mintjárára lettek létrehozva. Az emberi idegrendszer működését az azt alkotó idegsejtek (neuronok) mint információ-feldolgozó egységek és a köztük lévő kapcsolatok határozzák meg.

A számítógéptudomány azon területét, mely a fentiekben vázolt modellek létrehozásával foglalkozik gépi tanulásnak nevezik. A tanulás célja meglévő adatok alapján olyan minták, összefüggések felfedezése, amelyek segítségével ismeretlen adatokról is (közelítőleg) helyes következtetéseket lehet levonni.

Fontos különbség más tanításnak tekinthető programozási eljárásokkal szemben, hogy a neurális hálózatok esetében nem a probblémához illeszkedő matematikai algoritmus kódolása (procedurális programozás) vagy modell megalkotása (deklaratív programozás) történik, hanem csak a bemenő (input) és a kimenő (output) adatok figyelembevételével kerülnek beállításra a szükséges paraméterek. Ez azért lényeges, mert így a hálózat általános felépítésű, azaz széles körben alkalmazható, hiszen a paraméterek különböző beállításaival különböző konkrét feladatokra lehet alkalmazni.

A neurális hálózatok alapvető műveleti eleme a mesterséges neuron, amely a következőképpen definiálható.

2.1. Definíció. *Egy $f\colon \mathbb{R}^m \to \mathbb{R}$ leképezést (mesterséges) neuronnak nevezzük és felírható*

$$f(p) = \varphi(\sum_{i=1}^{m} w_i p_i + b)$$

alakban, ahol $\boldsymbol{p} \in \mathbb{R}^m$ egy bemeneti vektor, $\boldsymbol{w} \in \mathbb{R}^m$ egy súlyvektor, $b \in \mathbb{R}$ az eltolás (bias), és $\varphi\colon \mathbb{R} \to \mathbb{R}$ egy adott aktivációs függvény.

A hálózat szervezésének fontos része a különböző neuronrétegek (bemeneti, kimeneti, rejtett) kialakítása.

2.2. Definíció. *Egy neuronréteghez tartozó súlymátrix:*

$$W = \begin{bmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,n} \\ w_{2,1} & w_{2,2} & \dots & w_{2,n} \\ \vdots & \vdots & & \vdots \\ w_{m,1} & w_{m,2} & \dots & w_{m,n} \end{bmatrix}$$

alakban, ahol a sorok felelnek meg az egyes bemeneteknek, az oszlopok pedig a réteget alkotó neuronoknak.

Például $w_{3,2}$ értéke a második bemenet és a harmadik neuron közti kapcsolat súlyának az értékével egyenlő.

Ennek segítségével fogalmazható meg a következő meghatározás.

2.3. Definíció. *Egy neuronréteg kimenetének kiszámítása:*

$$a = \varphi(\boldsymbol{W}p + b)$$

ahol $\boldsymbol{p} \in \mathbb{R}^m$ *egy bemeneti vektor,* $\boldsymbol{W} \in \mathbb{R}^{m \times n}$ *egy súlymátrix,* $b \in \mathbb{R}$ *az eltolás (bias), és* $\varphi \colon \mathbb{R} \to \mathbb{R}$ *egy adott aktivációs függvény.*

A legegyszerűbb neurális hálózatnak a perceptron tekinthető, amely egyetlen neuronból valamint az ahhoz tartozó súlyok és eltolás összességéből áll. Ez azonban a valóságban csak szűk körben alkalmazható, viszont a korábban definiált neuronrétegek bevezetésével létrejött az úgynevezett többrétegű perceptron (Multi Layer Perceptron, MLP).

A feladat megoldása során fontos szerepe van az aktivációs függvény megválasztásának, ezzel befolyásolható a hálózat képessége és teljesítménye is. A rejtett rétegek általában ugyanazt az aktivációs függvényt használják, a kimeneti réteg viszont jellemzően ettől eltérőt, amely a feladat típusától függ.

Példák:

- ReLU:
$$\varphi(x) = max\{x; 0\} \quad x \in \mathbb{R}$$
Ha $x < 0$ akkor a visszaadott érték 0 lesz, különben x.

- Szigmoid vagy logisztikus függvény:
$$\varphi(x) = \sigma(x) = \frac{1}{1 + e^{-Kx}} \quad x \in \mathbb{R},\ K > 0 \text{ konstans}$$
Bármilyen x esetén a visszaadott érték $[0; 1]$ intervallumba fog esni úgy, hogy minél nagyobb x annál közelebb kerül 1-hez, és minél kisebb, annál közelebb 0-hoz.

- Tangens hiperbolikusz függvény:
$$\varphi(x) = tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \quad x \in \mathbb{R}$$
Bármilyen x esetén a visszaadott érték $[-1; 1]$ intervallumba fog esni úgy, hogy minél nagyobb x annál közelebb kerül 1-hez, és minél kisebb, annál közelebb -1-hez.

A különböző paraméterek meghatározása jelenti a hálózat tanítását. Konkrét célfüggvény nem áll rendelkezésre csak egy minta adathalmaz összetartozó bemenetekkel és kimenetekkel. De jól választott hibafüggvény segítségével meghatározható a modell álltal kapott kimenetek és az elvárt kimenetek távolsága, így lehetséges közelíteni a célfüggvényt a paraméterek változtatásával. Tehát a hibafüggvény minimalizálása a cél.

Példák:

- Átlagos négyzetes hiba (Mean Square Error/Quadratic Loss/L2 Loss):
$$L(y_i; \hat{y}_i) = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}$$

- Átlagos abszolút hiba (Mean Absolute Error/L1 Loss):
$$L(y_i; \hat{y}_i) = \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{n}$$

- Cross Entropy Loss/Negative Log Likelihood
$$L(y_i; \hat{y}_i) = -\sum_{i=1}^{m} y_i log(\hat{y}_i)$$

3. fejezet

Mély tanulás

A mély tanulás a gépi tanulás egy speciális fajtája. Az elnevezés utal a működésére, hiszen olyan neurális hálózatok használatát jelenti melyek több rejtett réteggel rendelkeznek. A fejezet megírásához az [1], [2], [6], [7], [9], [10] forrásokat használtam fel.

3.1. Előrecsatolt neurális hálózatok

A legegyszerűbb mély neurális hálózatok az előrecsatolt neurális hálózatok, amelyekben az egyes rétegeket az jellemzi, hogy az abban szereplő neuronok csak a közvetlen megelőző réteg neuronjaitól kapnak információt. Más szóval hurokmentes hálózatokról van szó. Az előzőekben meghatározott neuronréteg definíciója alapján az összekapcsolt rétegek hálózatát felírhatjuk az alábbi módon.

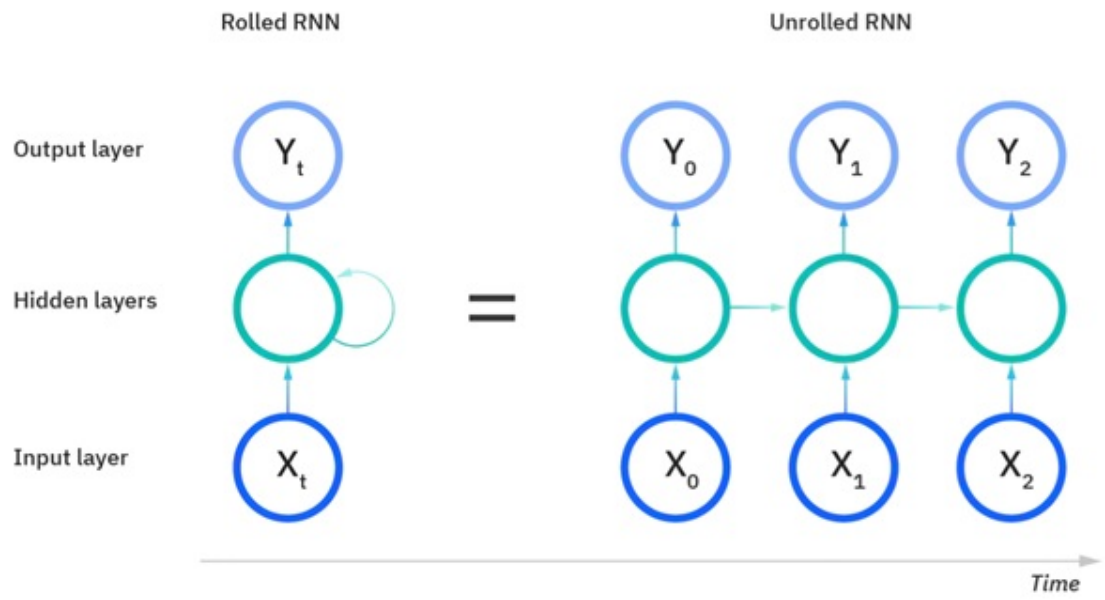
3.1. Definíció. *Előrecsatolt neurális hálózat:*

$$\phi_\theta = \phi_L \circ \cdots \circ \phi_2 \circ \phi_1(x)$$

ahol $\theta = (\theta_1, \theta_2, \ldots, \theta_K)$ *egy paramétereket tartalmazó vektor és* ϕ_i $(i = 1 \ldots L)$ *az egyes rétegekhez tartozó függvények.*

3.2. Visszacsatolt neurális hálózatok

Bizonyos problémák megoldására az előrecsatolt hálózatok nem alkalmasak. Hiszen az emberi agy is úgy működik, hogy gondolkodás közben a már megszerzett ismereteire is támaszkodik. Például egy könyvben olvasottak megértéséhez hozzájárulnak az előtte értelmezett szövegek is, így lehetséges kiválasztani többértelmű kifejezések különböző jelentései közül az odaillőt. A hagyományos neurális hálózatok legnagyobb hátránya, hogy nem képesek ezt a struktúrát megvalósítani, ezzel szemben a visszacsatolt (rekurrens) neurális hálózatok (RNN) alkalmasak e probléma kiküszöbölésére. Az információmegmaradást hurkok segítségével teszik lehetővé.

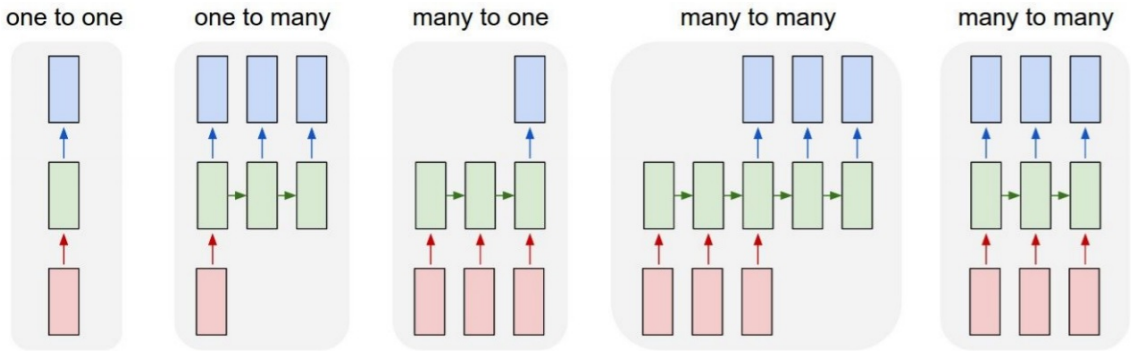


3.1. ábra. RNN hálózat szemléltetése többféle módon [6]

Az ábrán (3.1. ábra) látható baloldali ábrázolási mód a hálózatot teljes egészében szemlélteti, míg a jobboldali az egyes időbeli lépéseket is ábrázolja. Az utóbbin látható láncszerű szerkezet sajátosságából adódóan az RNN hálózatok szekvenciákra és listákra épülő problémák megoldása során használhatóak leghatékonyabban. Ilyen feladat például a beszédfelismerés, a nyelvi modellezés, az idegennyelvről fordítás és a képfeldolgozás.

Fontos megjegyezni, hogy a fenti ábra csak szemléltetésre szolgál, ami azt jelenti, hogy az RNN hálózatok felépítése nincs a fenti módon korlátozva, hanem a felhasználási területtől függően különböző típusai léteznek.

- One-to-many: Ezt a típust, akkor érdemes használni, ha egyetlen megfigyeléssel rendelkezünk, de egy szekvenciát szeretnénk ehhez kimenetként létrehozni. Például egy kép leírása különböző szavakkal.
- Many-to-one: Egy adott szekvencia feldolgozására és egyetlen kategóriába sorolására alkalmas. Például egy mondat hangulatának eldöntése, a közösségi felületek kommentjeinek vizsgálata és pozitív/negatív reakciók megállapítása.
- Many-to-many(different): A bemenetként kapott szekvencia egy másik szekvenciává átalakításához használható, ez lényegében fordításnak nevezhető. A két szekvencia hosszúsága tetszőleges és különböző is lehet.
- Many-to-many(same): A szekvencia minden megfigyeléséhez egy kimenet társítása a cél. Például egy videó képkockáihoz hozzárendelhető a képeken történő tevékenység, de ennek a típusnak a segítségével generálható zenemű is.



3.2. ábra. Különböző típusú RNN hálózatok [7]

3.2.1. BRNN hálózatok

Bizonyos esetekben problémát jelenthet, ha egy szekvencia elemzése során az aktuális állapot meghatározásához csak a korábbi megfigyeléseket lehetséges felhasználni. A következő mondatrész alapján nem egyértelműen eldönthető, hogy az *alma* szó tulajdonnevet jelöl-e.

„Nem hiszem el, hogy almát..."

Azonban a mondat további részét megvizsgálva egyértelművé válik.

„...kirúgták a munkahelyéről."

„...szedtél januárban."

A BRNN (Bidirectional recurrent neural networks) hálózatok képesek a jövőbeni adatokat is figyelembe venni. Az egyes szavak pontos vizsgálata céljából párhuzamosan vizsgálja a szekvenciát két irányból (a példában a mondat szavait balról-jobbra és jobbról-balra) oly módon, hogy a szekvencia bármely pontján ismerje az előtte és az utána álló elemeket is.

3.2.2. LSTM hálózatok

A hagyományos RNN hálózatok másik gyengesége, hogy azonos fontossággal kezelnek minden adatot, ami azt jelenti, hogy az új információk felülírják a régebbiket.

„A zebra szőre fekete és ...”

Egyéb információk nélkül is megjósolható, hogy a mondat következő szava a *fehér* lesz.

„Franciaországban nőttem fel, de most Bostonban élek. Anyanyelvi szinten beszélek...."

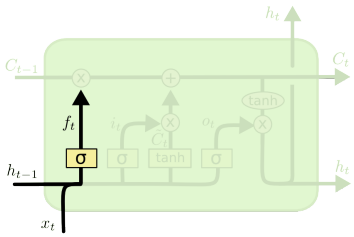
Ebben az esetben, annyi derül ki egyértelműen, hogy egy nyelv lesz a keresett kifejezés. Azonban távolabbi múltbeli információk is szükségesek lennének ahhoz, hogy a lehetőségek száma lecsökkenjen.

Az LSTM (Long short-term memory) hálózatok lényege, hogy az információ egy cella-állapot mechanizmuson halad keresztül, aminek hatására az adatok megjegyzése vagy elfelejtése szelektíven történik. Leggyakrabban egy futószalaghoz szokták hasonlítani ezt a hálózatot, mert egy szállítószalag feladata az, hogy a termékeket a különböző munkafázisokhoz szállítsa, ahol változtatásokat hajthatnak végre rajtuk. Az LSTM ugyanezzel a mechanizmussal működik. Az egyes módosítási pontokon különböző változtatásokat hajthatunk végre a beérkező adaton, szerkeszthetjük, törölhetjük, de új információt is

adhatunk a hálózatba. Három úgynevezett kapu szabályozza az információáramlást: a bemeneti, a kimeneti és a felejtésért felelős kapu.

Az első kapu (forget gate) szűri ki az elfelejteni való információkat egy szigmoid függvény segítségével. Jelen esetben a 0 a teljes elfelejtést, az 1 pedig a mindenre emlékezést jelenti.

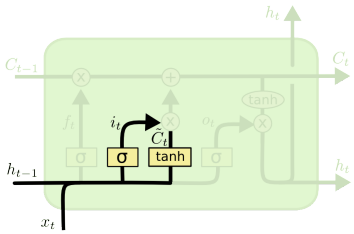
$$f_t = \varphi(W_f * [h_{t-1;x_t}] + b_f)$$



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

3.3. ábra. Felejtés kapu. [2]

A bemeneti kapu (input gate) frissíti a cellaállapotot, amihez szigmoid függvényt (0 a nem fontos, az 1 pedig a fontos) és tanh függvényt használ. A két függvény eredményét összeszorozza, azaz ebben a lépésben fog eldőlni, hogy mely információk maradjanak meg a tanh kiemenetéből.

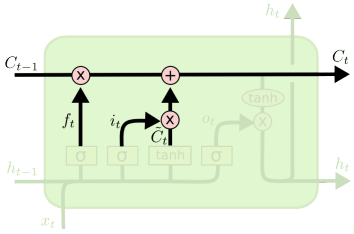


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

3.4. ábra. Bemeneti kapu. [2]

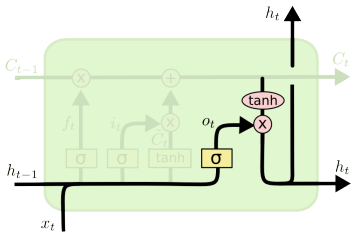
Ennek a két kapunak a segítségével már ki lehet számítani az új cellaállapotot, az aktuális cellaállapot és a felejtésre szolgáló kapu kimenete alapján. Így lehet eldobni a fölösleges információkat és frissíteni a cellaállapotot a releváns értékekkel.

Utolsó lépésként a kimeneti kapu meghatározza a rejtett állapotot. A korábbiakhoz hasonlóan az előző rejtett állapot és az aktuális bemenet adatai egy szigmoid függvény segítségével kerülnek feldolgozásra, emellett pedig az új sejtállapot értékét egy tanh függvény szabályozza. A kettőt összeszorozva megkapjuk a következő rejtett állapot értékét. Az új cellaállapot és az új rejtett állapot adódik tovább a következő fázisba.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

3.5. ábra. Az új cellaállapot meghatározása. [2]

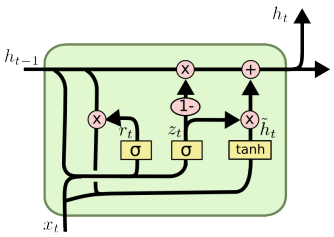


$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

3.6. ábra. Kimeneti kapu

3.2.3. GRU hálózatok

A GRU (Gated recurrent units) hálózatok nagyon hasonlítanak az LSTM hálózatokhoz. Azonban a GRU-k nem használnak cellaállapotot, hanem a rejtett állapotot használják az információk továbbítására. Valamint csak két kapu segítségével működnek: egy visszaállító kapu (reset gate) és egy frissítő kapu (update gate).



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

3.7. ábra. GRU. [8]

3.3. Transfer learning

A későbbiekben bemutatott feladat megoldása során felmerült a transfer learning módszer használata a feladathoz használt adathalmaz sajátosságai miatt. A módszer bemutatásához a [2] forrást használtam fel.

Osztályozási feladatok során a gépi tanulás hagyományos módszerei hasonló elven működnek abból a szempontból, hogy egy jól működő modell létrehozásához nagyszámú tanulóadat szükséges. Ha ez nem áll rendelkezésre vagy nincs elegendő erőforrás a tanításhoz, akkor a transfer learning módszere alkalmazható, amelynek lényege a már betanított hálók újra felhasználása. Ezzel a metódussal egy korábban betíntott háló továbbtanítható.

A valóságban transfer tanulásnak feleltethető meg például bizonyos hangszertanulási folyamatok egymásutánisága, aki orgonán megtanul játszani, az később könnyebben tanul zongorázni.

Az eljáráshoz tartozó legfontosabb fogalmak a tartomány (domain) és a feladat (task).

- Tartományt egy χ (feature space), és egy ehhez tartozó $P(X)$ eloszlás határoz meg. Két tartományt különbözőnek tekintünk, ha eloszlásukban különböznek. Jelölése: $\mathcal{D} = \{\chi, P(X)\}$, ahol $X = \{x_1, \ldots, x_n\} \in \chi$.
- Adott tartományhoz tartozó feladat egy γ célváltozó, és egy $f(.)$ predikciós függvény segíségével definiálható. Jelölése: $\mathcal{T} = \{\gamma, f(.)\}$.

Adott egy forrástartomány és egy forrásfeladat, ezeket jelölje $\mathcal{D}_S$ és $\mathcal{T}_S$, valamint egy céltartomány és célfeladat, $\mathcal{D}_T$ és $\mathcal{T}_T$. Ekkor a transfer learning módszer célja $f_{\mathcal{T}}(.)$ tanulását segíteni $\mathcal{D}_T$ céltartományban, $\mathcal{D}_S$ forrástartomány és $\mathcal{T}_T$ forrásfeladat ismeretinek felhasználásával, ha $\mathcal{D}_S \neq \mathcal{D}_T$ és $\mathcal{T}_S \neq \mathcal{T}_T$.

4. fejezet

Feladat és megoldása

4.1. Fehérjecsalád klasszifikáció

A hasonló működésű fehérjék csoportosíthatóak, ezeket a csoportokat nevezik fehérjecsaládoknak. Ilyen családba sorolásra példa a fentebb bemutatott PFAM osztályozás. Alapvető biológia feladat a fehérjék működésének meghatározása aminosav-szekvenciájuk segítségével. Ennek megoldásához a gépi tanulás módszerei alkalmazhatóak. Neurális hálók segítségével vizsgálható a fehérjék elsődleges szerkezete, a fehérjeszekvencia és adott osztályba sorolásuk közötti kapcsolat. Ezáltal olyan minta nyerhető ki, melynek segítségével besorolatlan fehérjék is a már meglévő osztályokba sorolhatók, így információhoz lehet jutni a működésüket illetően.

Szakdolgozatomban a fehérjék PFAM osztályozása során meghatározott családokba sorolásával dolgoztam.

4.2. Feladathoz használt adathalmazok

Összesen három adathalmazt használtam a feladat megoldása során.

4.2.1. Kaggle forrás

A Kaggle egy online közösségi felület, amely adattudomány valamint a gépi tanulás témakörére épül. A 2010-ben indult weboldal eredetileg gépi tanuláshoz kapcsolódó versenyeket

kínált a felhasználók számára. Mára a kibővült oldalon számos adathalmaz és a felhasználók által publikált kód, modell is megtalálható.

Az általam használt adathalmaz elérhetősége:
`https://www.kaggle.com/googleai/pfam-seed-random-split`

Ez az adatkészlet a fehérjék osztályozásához készült a szerkezetükben fellelhető domének alapján. A domén a fehérjeszekvencia funkcionálisan elkülönölő alszekvenciája, és általában a domének egy-egy konkrét funkciót határoznak meg. Minden PFAM család tartalmaz egy úgynevezett "seed" csoportot, amelynek tagjai manuálisan lettek besorolva többszöri vizsgálat után, és ezen csoport segítségéve határozzák meg számítógépes módszerekkel a családok további tagjait. Ez az adathalmaz csak a seed csoportba tartozó fehérjéket és azok domén adatait tartalmazza. Körülbelül összesen 1 millió rekordból alkotja, melyek 18 ezer családból kerültek ki.

Egy rekord a következőképpen épül fel:

```
sequence: HWLQMRDSMNTYNNMVNRCFATCIRSFQEKKVNAEEMDCTKRCVTKFVGYSQRVALRFAE
family_accession: PF02953.15
sequence_name: C5K6N5_PERM5/28-87
aligned_sequence: ....HWLQMRDSMNTYNNMVNRCFATCI...........RS.F....QEKKVNAEE.....MDCT....KRCVTKFVGYSQRVALRFAE
family_id: zf-Tim10_DDP
```

4.1. ábra. Kaggle forrásból. [14]

- sequence: Domén aminosav-szekvenciája láncba rendezve, általában ezek alkotják a modell bemenetét.
- family_accession: PFAM családot meghatározó PFXXXXX.YY formában megadott kód, melyben PFXXXXX a család azonosítója, YY pedig a verziószám, amely a PFAM adatbázis folyamatos frissítéséből adódik. Általában ezek lesznek a modell címkéi.
- sequence_name: Szekvencia egyedi neve meghatározott formában.
- aligned_sequence: A fehérjeszekvenciában megtalálható domének, üresen hagyva a doménen kívűli aminosavak helyét.
- family_id: PFAM család elnevezése.

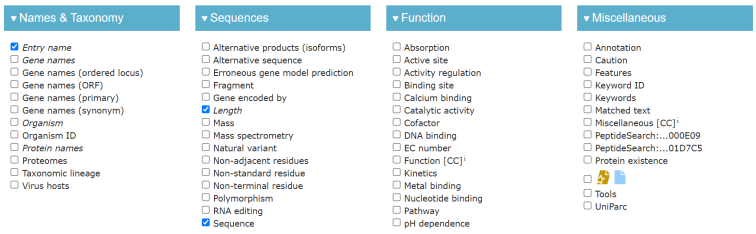
Eredeti formában az adatok már három részre vannak osztva; tanuló, validációs és teszt halmazba véletszerűen történt a besorolásuk, rendre 80-10-10 százalékos arányban. Az adatok .csv kiterjesztésű fájlokban találhatóak.

4.2.2. UniProt forrás

A fentebb bemutatott UniProt fehérjeadatbázisból az UniProtKB/Swiss-Prot adathalmazt használtam fel.

UniProtKB/Swiss-Prot online elérhetősége:
`https://www.uniprot.org/uniprot/?query=reviewed:yes`

Az adathalmaz összesen 564 638 rekordot tartalmaz, mindegyikéhez információs mezők tartoznak, melyek megjelenítése választható a felhasználó széndáka szerint. A teljesség igénye nélkül az alábbi ábrán (4.2. ábra) látható néhány mező.



4.2. ábra. UniProt forrásból. [13]

Az általam kiválasztott mezőket tartalmazó rekordok a következőképpen épülnek fel:

- Entry: UniProt azonosító
- Entry name: Szekvencia neve
- Length: Szekvencia hossza
- Sequence: Teljes szekvencia
- Cross-reference (Pfam): PFAM családot meghatározó, PFXXXXX formában megadott azonosító

A letöltés során a fájl kiterjesztése választható, számomra a .tab kiterjesztés volt az ideális.

4.2.3. PFAM forrás

A feladat megoldása során szükségessé vált olyan adathalmaz használata, mely az előzőekhez hasonlóan nagyszámú adatot tartalmaz, de azoktól eltérően az egyes családokat is nagyszámban és kiegyenlítetten képviseli. Ennek érdekében sajátkezűleg hoztam létre egy adathalmazt úgy, hogy véletszerűen választottam ki a különböző családok közül olyanokat, melyek teljes számossága 5 000 - 13 000 közé esett.

Az így létrejött adathalmaz összesen 270 osztályt képvisel és 1 963 245 rekordot tartalmaz, egyelőre még kiegyenlítetlenül.

Az egyes családba tartozó fehérjéket tartalmazó fájlok FASTA formátumban voltak elérhetőek. Ez egy olyan szöveges formátum, mely nukleotid- vagy fehérjeszekvenciák tárolására szolgál.

Példa egy ilyen fájlban tárolt adatokra:

```
>A0A1A8W3S0_PLAMA (A0A1A8W3S0)
MEKSLDLSNFNKNDREKIMRKINKAEYEDTMTTYNSIVERCFNECITSFRSKELDSNEN
NCILNCVKKFSIFSQRIGMKFTQNLNSEMQKKS
>A0A0L7QPE3_9HYME (A0A0L7QPE3)
MAMQIPTNVDPEQIKSVRDFVASYNKLIHTCFTDCINEFTSRDVQAKEETCALNCMEKY
LKMNQRISQRFEEFQMLANENALAAQKKISEAKK
>A0A074SAC1_9AGAM (A0A074SAC1)
MDTNQKFDEATQRELQTFIEQEQAKARVQATTHQLTDMCWTKCITGSVSTRFSSSEAYC
LQNCVDRFLDTSLFIVKKLDEQRRSMQGGQ
>G3XAN8_HUMAN (G3XAN8)
MRKHSCRKVASLRRTMAELGEADEAELQRLVAAEQQKAQFTAQVHHFMELCWDKCVEKP
GNRLDSRTENCLSSCVDRFIDTTLAITSRFAQIVQKGGQ
>G3Q3E1_GASAC (G3Q3E1)
AVCEQRSSADMDPVKAQQLAAELEVEMMADMYNRMTNACHRKCVPPHYKEAELTKGESV
CLDRCVAKYLDLHERLGRKLTELSVQDEETMRKASLGSG
```

4.3. ábra. PFAM forrásból.

Minden rekord > szimbólummal kezdődik, melyet a szekvencia azonosítója követ. A következő sorokban a szekvencia következik. Úgy ajánlott eltárolni a hosszú szekvenciákat, hogy az egyes sorokba maximum 80 karakter kerüljön. Minden fájl neve tartalmazza a PFAM család azonosítót PFXXXXX formátumban, melyet majd a későbbiek, az adatok feldolgozása során fogok felhasználni.

4.3. Adatok előkészítése és alapstatisztikák

Kaggle forrás:
Az adatok későbbi feldolgozásához megfelelőbbnek találtam az eredetileg már felosztott adathalmaz újra egyesítését, így a modelleknek megfelelő, későbbiekben tárgyalt átalakítást egy lépésben el tudtam végezni. Második lépésben hoztam létre a tanításhoz, validáláshoz és teszteléshez szükséges halmazokat.

A forrásfájlok formátuma lehetővé tette, hogy beolvasás után a adatokat egy DataFrameben tudjam tárolni, ez a feladathoz legmegfelelőbb objektum, ezért ezzel kapcsolatban más átalakítást nem kellett végeznem. Az alapstatisztikák elvégzéséhez egy rövid algoritmussal minden sort kiegészítettem az adott szekvencia hosszát meghatározó értékkel.

UniProt forrás:
Az adatok egyetlen forrásfájlban való tárolása lehetővé tette a DataFrame objektumba való beolvasást. A létrehozott táblát megvizsgálva olyan rekordokat találtam, melyek egynél több PFAM családazonosítót tartalmaztak, így ezek kiszűrésére volt szükség. Ezek eltávolítása után 351 459 db rekord maradt.

PFAM forrás:
Mivel a családokhoz tartozó adatokat egyesével kellett letöltenem, ezért összesen 270 db fájl állt rendelkezésemre. FASTA formátumú fájlok kezelésére Python programozási környezetben a biopython csomag szükséges, melyet kifejezetten erre a környezetre fejlesztettek ki bioinfomatikai feladatokhoz. Létrehoztam egy automatizált algoritmust, mely ötösével olvasta be a FASTA fájlokat. Betöltötte az adatokat egy DataFrame objektumba, majd .csv formátumban mentette az adathalmazt. Az így létrejött 54 db fájlt egy egyszerűbb algoritmus segítségével tudtam egyesíteni egy DataFrame objektumban. Erre a megoldásra a rendelkezésre álló erőforrások korlátai miatt volt szükség, és így tudtam optimális sebességet elérni az adatok beolvasásához. Az alapstatisztikák elvégzéséhez ez esetben is minden sort kiegészítettem az adott szekvencia hosszát meghatározó értékkel.

Az előzőekben tárgyalt előkészítési folyamatok az adatok kezelését segítették elő, de az adott feladat megoldásához további vizsgálatokra van szükség, ezeket a vizsgálatokat nevezem alapstatisztikáknak. Adatbányászati szempontból az adatfeltárás folyamatának részét képezik. Lényege olyan statisztikai és vizuális elemzés készítése, amelyek elősegítik az adatok megismerését, rávilágíthatnak az esetleges hibákra (más szóval zajokra), és segítségükkel könnyebben kiválasztható a feladat megoldásához szükséges módszer is. Szerepük

nem elhanyagolható, mert hibás adathalmazon egy jó modell nem lesz képes megfelelően működni.

Egy beépített függvény segítségével összegzés kapható a DataFrame objektumban tárolt adathalmazról oszlopok szerinti bontásban.

```
data_kaggle.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 1339083 entries, 0 to 13554
Data columns (total 6 columns):
 #   Column            Non-Null Count    Dtype
---  ------            --------------    -----
 0   family_id         1339083 non-null  object
 1   sequence_name     1339083 non-null  object
 2   family_accession  1339083 non-null  object
 3   aligned_sequence  1339083 non-null  object
 4   sequence          1339083 non-null  object
 5   seq_char_count    1339083 non-null  int64
dtypes: int64(1), object(5)
memory usage: 71.5+ MB
```

```
data_uniprot.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 351460 entries, 0 to 351459
Data columns (total 7 columns):
 #   Column                   Non-Null Count   Dtype
---  ------                   --------------   -----
 0   index                    351460 non-null  int64
 1   Entry                    351460 non-null  object
 2   Entry name               351460 non-null  object
 3   Length                   351460 non-null  int64
 4   Sequence                 351460 non-null  object
 5   Cross-reference (Pfam)   351460 non-null  object
 6   char_count               351460 non-null  int64
dtypes: int64(3), object(4)
memory usage: 18.8+ MB
```

```
data_pfam.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 1963245 entries, 0 to 1963244
Data columns (total 4 columns):
 #   Column          Dtype
---  ------          -----
 0   ID              object
 1   SEQ             object
 2   PFAM            object
 3   seq_char_count  int64
dtypes: int64(1), object(3)
memory usage: 74.9+ MB
```

4.4. ábra. Információs tábla.

A PFAM forráshoz tartozó összegzés más formában jelent meg, mint a többi adatforráshoz tartozó. Először arra következtettem, hogy nem megfelelően hoztam létre a táblazatot, és ez okozza az eltérést. Vizsgálatok során megállapítottam, hogy az adatok nagy száma miatt történt, ezért olyan információs táblákat készítettem, melyek két részletben összegzik az adathalmazt. Erre azért volt szükség, hogy információt kapjak az üres mezőket tartalmazó rekordok létezéséről és ha szükséges kiszűrhessem azokat. Nem találtam ilyen rekordot egyik adathalmazban sem.

```
data_pfam.loc[0:999999].info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000000 entries, 0 to 999999
Data columns (total 4 columns):
 #   Column          Non-Null Count    Dtype
---  ------          --------------    -----
 0   ID              1000000 non-null  object
 1   SEQ             1000000 non-null  object
 2   PFAM            1000000 non-null  object
 3   seq_char_count  1000000 non-null  int64
dtypes: int64(1), object(3)
memory usage: 38.1+ MB
```

```
data_pfam.loc[1000000:1963244].info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 963245 entries, 1000000 to 1963244
Data columns (total 4 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   ID              963245 non-null  object
 1   SEQ             963245 non-null  object
 2   PFAM            963245 non-null  object
 3   seq_char_count  963245 non-null  int64
dtypes: int64(1), object(3)
memory usage: 36.7+ MB
```

4.5. ábra. Információs tábla két részletben.

Klasszifikációs feladatokhoz használt adathalmaz esetében fontos, hogy az adatok kiegyenlítetten képviseljék az egyes osztályokat, ezért meg kell vizsgálni azok számosságát. Ehhez egy beépített függvényt használtam, mely csökkenő sorba rendezi az osztályokat számosságuk szerint.

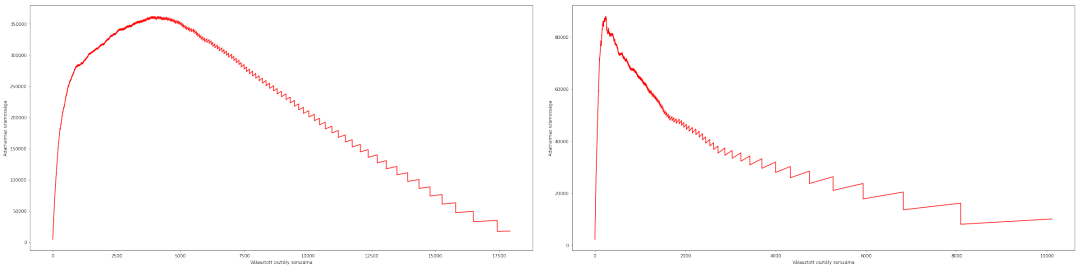
```
data_kaggle.groupby('family_accession').

family_accession
PF13649.6      4545
PF00560.33     2407
PF13508.7      2199
PF06580.13     1921
PF02397.16     1908
               ...
PF17574.2         1
PF17575.2         1
PF01445.17        1
PF17576.2         1
PF05550.11        1
Length: 17929, dtype: int64
```

```
data_uniprot.groupby('Cross-reference (Pfam)')

Cross-reference (Pfam)
PF00005;     2289
PF00069;     1864
PF00156;     1722
PF00001;     1651
PF00067;     1554
             ...
PF09440;        1
PF09432;        1
PF09386;        1
PF09385;        1
PF07305;        1
Length: 10113, dtype: int64
```

```
data_pfam.groupby('PFAM')

PFAM
PF04607    14739
PF01029    13762
PF16124    13315
PF01421    13086
PF00297    12852
           ...
PF08378     5052
PF01087     5046
PF01693     5018
PF10358     5006
PF02790     5002
Length: 270, dtype: int64
```

4.6. ábra. Osztályok számossága csökkenő sorrendben.

A PFAM forrásból származó adathalmaz esetén egyértelműen megállapítható, hogy az osztályok számosságát a legkisebb osztály méretére korlátozni megfelelő lesz. A másik két adatforrás esetében további vizsgálatok szükségesek. Diagramot készítettem, mely a teljes adathalmaz számosságának változását szemlélteti, annak függvényében, hogy a rendezett osztályok közül sorban hány darabot választok. A kapott értékek azt mutatják meg, hogy mekkora lesz az adathalmaz mérete ha a választott n osztály számosságát korlátozom a legkisebb osztály számosságára.



4.7. ábra. Adathalmaz számosságának változása.

Az egyes szekvenciák hossza valamint a szekvenciákat alkotó karakterek előfordulási száma is fontos tényező lesz a feladat megoldása során ezért diagramon ábrázoltam ezeket az értékeket is.

Az alapstatisztikák elvégzése után elvégeztem a szükséges átalakításokat, és így alakultak ki a végső adathalmazok. Az osztályok számát illetően a Kaggle adathalmazt ... darabra, az UniProt adathalmazt 243 darabra, a PFAM adathalmazt 1000 darabra korlátoztam. Az aminosavakat kódoló karakterek előfordulási számát tekintve megállapítható, hogy 20 darab fehérjealkotó aminosav meghatározó számban, 4 darab pedig elhanyagolható számban fordul elő.

A különböző neurális hálózatok meghatározott formában várják a bemenetet, ezért a felhasznált modelleknek megfelelően szükséges elkódolni az adatokat. Az általam használt modellekhez a szekvenciák numerikus vektorokba kódolására volt szükség. A szekvenciákat alkotó aminosavaknak megfelelő betűkészletet ABC sorrendbe rendeztem és azokat rendre 1-től 20-ig az egész számoknak feleltettem meg, a fentebb említett 4 darab aminosav a 0 kódot kapja. A sorrend kialakítása egyéni döntés kérdése és nem fogja befolyásolni a modell működését.

A következő művelet a rendelkezésre álló erőforrások korlátai miatt lehet szükséges, segítségével a szekvenciák mérete korlátozható. Választható paraméterek, hogy a szekvencia eleje vagy vége kerüljön levágásra, mekkora legyen a maximális hossza a szekvenciáknak, és az olyan szekvenciák esetében, amelyek rövidebbek a megadott maximum értéknél, a kipótláshoz szükséges 0 karakterek a levágott rész elejére vagy végére kerüljenek. Az így létrejött vektorok lesznek a modellek bemenetei.

Szükséges még az osztályoknak megfelelő címkék kialakítása. Először

A cimkehalmaz az elérhető osztálycímkehalmaz alapján kerül meghatározásra, majd minden szekvenciához kiosztásra kerül a számként kódolt címke az eredeti osztályának megfelelően. Fontos, hogy az egyes osztályokat reprezentáló számok nagysága ne befolyásolja az eljárást, ezért one-hot vektorokká kell átalakítani azokat. A one-hot vektorok jellemzője, hogy egyetlen helyen 1 az értékük, mindehol máshol 0. Egy címkehalmaz és a hozzá tartozó one-hot vektorhalmaz között kölcsönösen egyértelmű megfeleltetés van létesítve, ebből következik, hogy a one-hot vektorok hosszát a címkehalmaz számossága határozza meg.

5. fejezet

Eredmények

5.0.1. LSTM/BiLSTM és GRU/BiGRU

Négy típusú réteget használtam a szekvenciák vizsgálatához, LSTM réteget és GRU réteget, valamint ezeknek a bidirectional, azaz kétirányú megfelelőjét. A korábban részletesebben bemutatott rétegek működése nagyon hasonló, ezért az adatok vizsgálata során ezeket a rétegeket felcseréltem egymással a különböző modellek megalkotásához, a háló struktúráján mást nem változtattam. Az ábrán látható két különböző konstrukció, melyek mindegyikéhez a fentiek értelmében négy modell tartozik.

Rétegek
InputLayer
Embedding
(Bi)LSTM/GRU
Dense

Rétegek
InputLayer
Embedding
(Bi)LSTM/GRU
Dropout
Dense

5.1. ábra. Modellek felépítése.

A továbbiakban az említett négy típusú rétegre RNN rétegként fogok hivatkozni, ha összeségében szeretnék róluk írni.

A korábbi vizsgálatok és az adathalmazok sajátosságai alapján, úgy ítéltem meg, hogy az UniProt forrásból származó adathalmaz a leginkább megfelelő arra, hogy segítségével meghatározzam a megfelelő paraméterbeállításokat. Két szempontot vettem figyelembe, amikor a három adathalmaz közül választottam. Az első szempont az adatok minpsége volt, és az UniProt forrás ellenőrzött adatokat tartalmaz. A második szempont az volt, hogy a lehető legáltalánosíthatóbb modellt kapjak, és ennek az adathalmaznak köszönhetően teljes szekvenciákkal tudtam dolgozni, nem csak az azokban szereplő doménekkel.

Kiindulásként 243 darab osztályra korlátoztam az adathalmazt, mely esetében minden osztály számossága 362 szekvencia lett. A tanító-, validációs, teszhalmaz számossága rendre 70372, 8797, 8797 volt. Minden szekvenciát 100 hosszúra vágtam a szekvencia elejétől indulva, és az ennél rövideb szekvenciák esetében a sor végét egészítettem ki 0 kóddal.

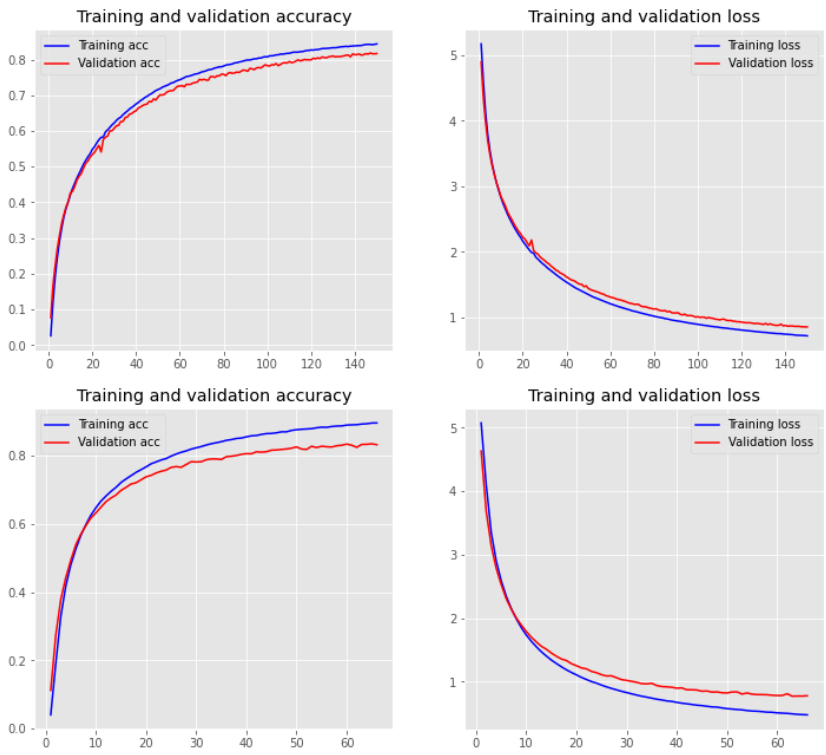
Az alapmodell RNN rétegének neuronszámát kiindulásként 32-nek választottam, de az eredményeket látva ezt 64-re növeltem. Az alábbi táblázatban szereplő tanítható paraméterszámokat kaptam az egyes modellekhez tartozóan. Kezdetben minden modell 50 epochon keresztül futtattam, a batch_size-t pedig 256-ra állítottam be. A kísérletekhez tartozó ábrákon megyfigyelhető, hogy az epochok számát szükség esetén növeltem. Erre leginkább a kezdeti vizsgálatok alkalmával volt szükség.

	Neurons	LSTM	BiLSTM	GRU	BiGRU
Total	32	31 315	56 699	26 259	49 587
params	64	67 891	132 851	55 731	108 531

Az alapmodell első módosítását a jobb pontosság elérése érdekében végeztem el, ez az elvárásaimnak megfelelően javította a modell eredményeit. A két táblázatot összehasonlítva észrevehető a kétirányú RNN rétegek jobb teljesítménye, de további vizsgálatok szükségesek ennek igazolására.

	LSTM	LSTM	GRU	GRU
Neurons	32	64	32	64
Train acc (%)	87.87	92.99	84.66	92.26
Val acc (%)	84.49	87.33	81.7	86.96
Test acc (%)	84.11	86.69	80.97	86.68

	BiLSTM	BiLSTM	BiGRU	BiGRU
Neurons	32	64	32	64
Train acc (%)	93.19	93.37	92.26	95.41
Val acc (%)	89	90.45	86.96	88.25
Test acc (%)	87.62	90.17	86.68	88.49



5.2. ábra. GRU 32 (felül) és 64 (alul) neuronnal tanítva.

Részletesebb bemutatásra a GRU hálózat grafikonjait választottam, mert ennél a modellnél tapasztaltam a legnagyobb eltérést a módosítás után. Bár a modellek egymáshoz közeli pontosságot produkálnak a valdidációs halmazon, az egyik lényeges különbség, hogy a 64 neuronnal ellátott modell kevesebb, mint feleannyi epoch alatt teljesíti ezt, és valamivel jobb értékeket is ér el. Viszont az is észrevehető a kék és piros grafikon egymással kölcsönös helyzetéből, hogy a túltanulás irányába mozdult el. A második megfigyelés mindegyik modell esetében megfigyelhető.

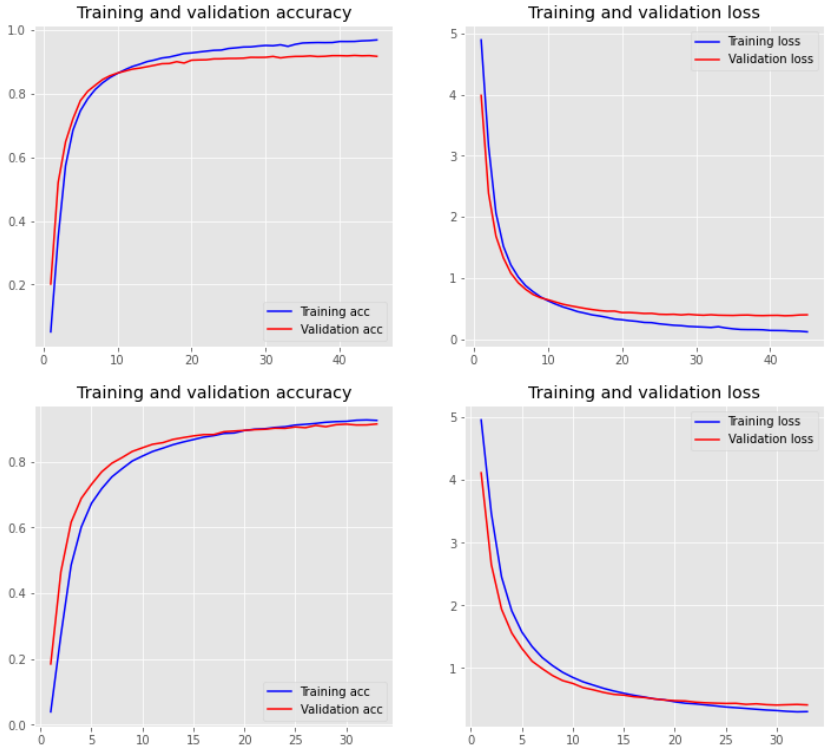
A következő kísérletek alkalmával tovább növeltem az RNN rétegek neuronjainak számát további javulást várva, azonban szükségesnek éreztem egy Dropout réteg használatát is. Ez a típusú réteg a túltanulás megakadályozására szolgál, olyan módon, hogy a megelőző réteg neuronjai közül véletlenszerűen kihagy meghatározott számú neuront. A Dropout réteg hiperparaméterét először 0.2-nek választottam, de bizonyos modellek esetében azt állapítottam meg, hogy további korlátozások szükségesek a túltanulás kockázata miatt, ezért további vizsgálatokat végeztem 0.4 paraméterű Dropout réteggel.

	neuronok	Dropout	LSTM	BiLSTM	GRU	BiGRU
Total	128	0.2	165 619	328 307	133 107	263 283
params	128	0.4	165 619	328 307	133 107	263 283

	LSTM	LSTM	GRU	GRU
RNN réteg (neuronok száma)	128	128	128	128
Dropout	0.2	0.4	0.2	0.4
Train acc (%)	98.45	95.94	94.75	95.85
Val acc (%)	91.66	91.49	88.21	89.47
Test acc (%)	91.29	90.67	87.52	89.09

	BiLSTM	BiLSTM	BiGRU	BiGRU
RNN réteg (neuronok száma)	128	128	128	128
Dropout	0.2	0.4	0.2	0.4
Train acc (%)	99.17	99.25	98.78	98.89
Val acc (%)	93.12	93.88	92.46	92.33
Test acc (%)	92.83	93.61	91.84	92.09

A táblázatok értékeit összehasonlítva nem figyelhető meg nagy eltérés a paraméter változtatásával, ezért az LSTM modell eredményeinek a grafikonján reprezentálom.



5.3. ábra. LSTM 0.2 (felül) és 0.4 (alul) paraméterű Dropout réteggel.

Az alsó ábrán jól megfigyelhető a kék tanulási görbe alapján, hogy a tanulásra miként hat a Dropout paraméterértékének növelése, hiszen továbbra is jó validációs pontosság mellett a tanulási görbe körülbelül 15 epochkal később éri csak el azt a szinten, ahol elkezd túltanulni. Emellett a 0.4-es Dropout réteget használó modell kevesebb epoch alatt éri el ugyanazt a pontosságot.

Az eredmények alapján más túltanulás megelőzésére alkalmas módszer kipróbálása mellett döntöttem a Dropout réteggel együtt alkalmazva.

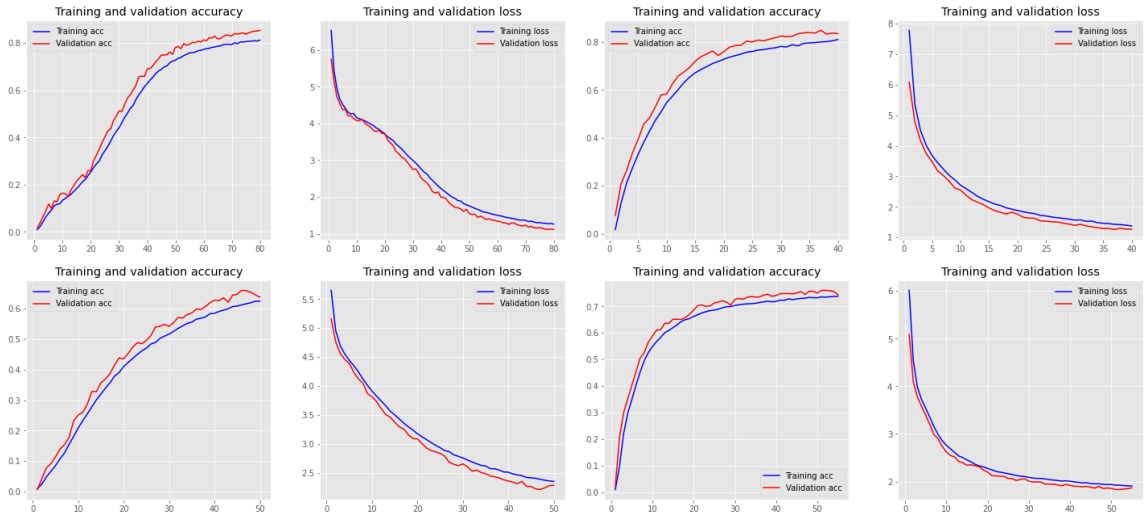
Az általam kipróbált regularizációs paraméterek:

- kernel_regularizer = l2(0.01)
- recurrent_regularizer = l2(0.01)
- bias_regularizer = l2(0.01)

Ez a módszer a hibafüggvényt módosítja, így próbálva megelőzni az esetlegesen magasan kiúgró súlyértékeket, tehát azt, hogy a hálózat a ritka mintákat túl fontosnak kezelje.

Először alacsony paraméterek beállításával próbálkoztam. A regularizáció hatására a modellek erőtelejsen rosszabb teljesítményt produkáltak, mint azt elvártam.

	LSTM	GRU	BiLSTM	BiGRU
RNN réteg (neuronok száma)	128	128	128	128
Dropout	0.4	0.4	0.4	0.4
Train acc (%)	87.02	65.47	85.27	76.34
Val acc (%)	85.43	63.82	83.45	74.38
Test acc (%)	85.3	64	84.47	75.22

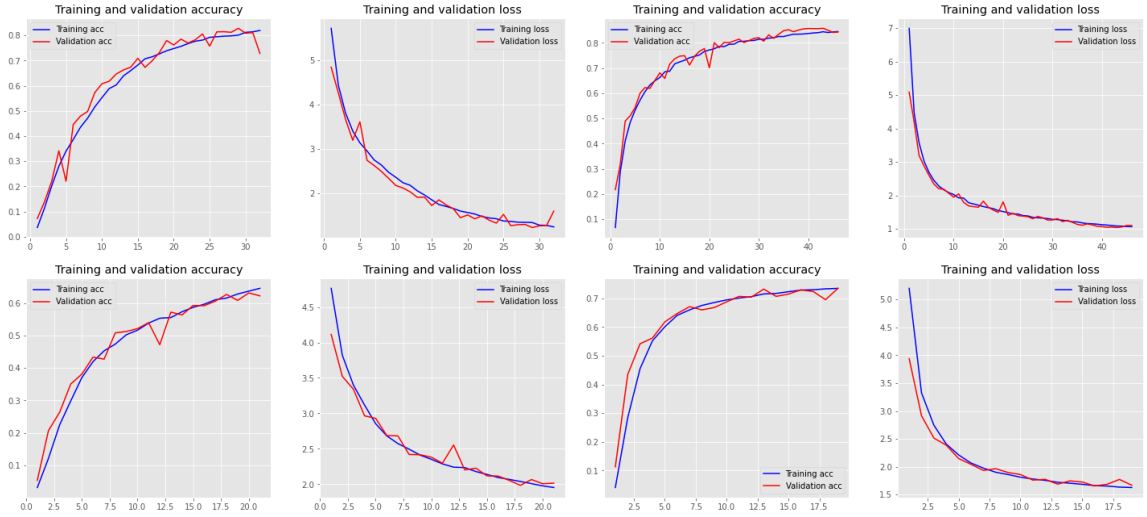


5.4. ábra. LSTM és BiLSTM (felül), GRU és BiGRU (alul)

A pontosság romlása mellett megfigyelhető, hogy a tanulás nem egyenletes, ami eddig nem volt jellemző a modellekre, ebből a szempontból a kétirányi RNN réteget használó modellek jobban teljesítettek a nem kétirányúakhoz képest. Ezenkívűl az eddig sima görbék kiugró értékeket mutatnak, ebből arra lehet következtetni, hogy a súlyok apró változtatásai nagy eltéréseket okoznak.

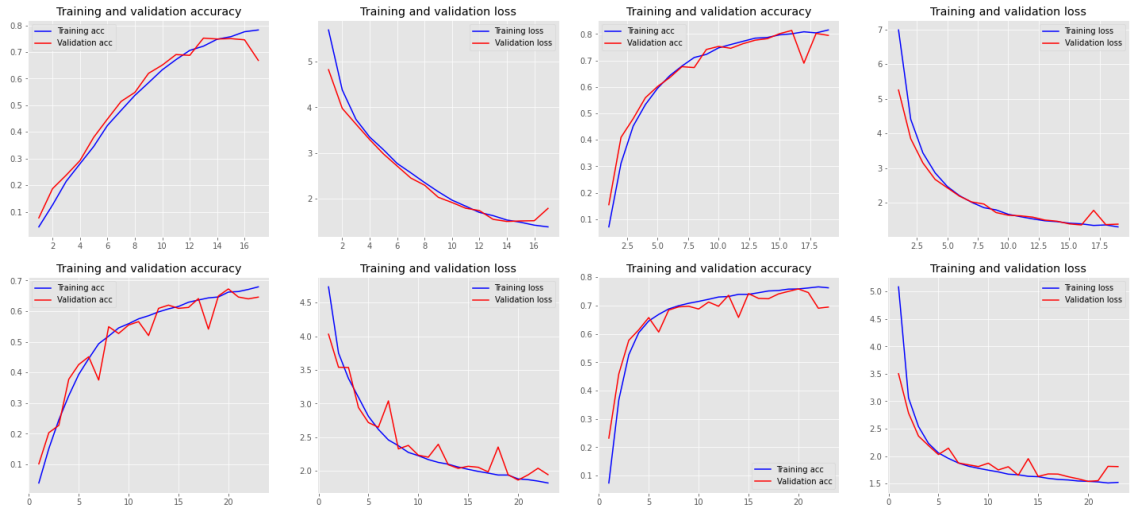
Arra a döntésre jutottam, hogy a tanítás során használt adathalmazt fogom módosítani, ezért csökkentettem az osztályok számát, aminek hatására kétszeresére nőtt az egyes osztályok számossága. A továbbiakban 100 darab, 701 számosságú osztállyal folytattam a kísérleteket.

Az alábbi ábrán látható a modellek teljesítménye a módosított adathalmazon, 128 neuronnal az RNN rétegen, 0.4-es Dropout réteggel, és a felsorolt regularizációkkal.



5.5. ábra. LSTM és BiLSTM (felül), GRU és BiGRU (alul)

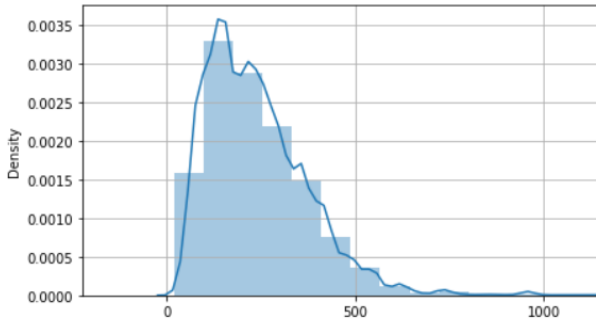
Az eredményeket nem találtam megfelelőnek, ezért a paramétert 0.2-re módosítottam.



5.6. ábra. LSTM és BiLSTM (felül), GRU és BiGRU (alul)

Végül úgy döntöttem, a további kíséleteket regularizációs paraméterek nélkül végzem.

Az eddigiekben legjobban teljesítő paraméterbeállításokat választottam, és arra a megállapításra jutottam, hogy megint az adatokon szükséges módosításokat végeznem. Az alapstatisztikák alkalmával megvizsgáltam az egyes szekvenciákhosszúságok előfordulásának grafikonjait. Ezek alapján jónak láttam hosszabb szekvenciákkal végezni kísérleteket. Az ábra alapján 300 és 500 hosszértékeket tartottam ideálisnak a kísérletek elvégzéséhez.



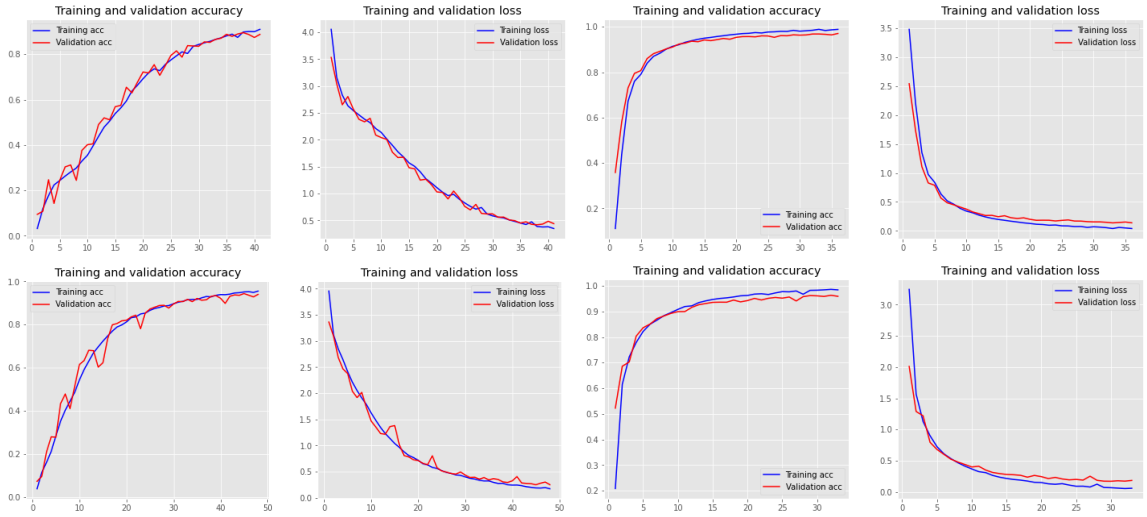
5.7. ábra. UniProt forrás

	LSTM	LSTM	LSTM	GRU	GRU	GRU
Length	100	300	500	100	300	500
Dropout	0.2	0.2	0.2	0.2	0.2	0.2
Train acc (%)	97.04	90.63	38.24	97.45	96.84	86.96
Val acc (%)	93	88.75	37.76	91.94	94.01	86.06
Test acc (%)	93.77	90.11	38.67	93.17	95.16	86.61

	BiLSTM	BiLSTM	BiLSTM	BiGRU	BiGRU	BiGRU
Length	100	300	500	100	300	500
Dropout	0.2	0.2	0.2	0.2	0.2	0.2
Train acc (%)	98.52	99.62	99.16	99.59	98.93	96.85
Val acc (%)	94.21	97.06	96.78	94.50	95.88	94.17
Test acc (%)	94.89	97.65	97.42	95.06	96.52	95.22

Végeredményben elmondható, hogy az LSTM vagy GRU réteget használó modellek végig rosszabbul teljesítettek, mint a kétirányú RNN réteget tartalmazó modellek, azonban az utolsó tesztek alkalmával már számottevő az eltérés. Ezek alapján már határozottan megállapítható, hogy az ilyenfajta elemzések során kétirányú rétegek alkalmazása lesz

a megfelelő, ezért a továbbiak csak a BiLSTM és BiGRU modelleket használtam, valamint úgy értékeltem, hogy a 300 hosszú szekvenciákat elemző modelljeim teljesítettek a legjobban ezen az adathalmazon.



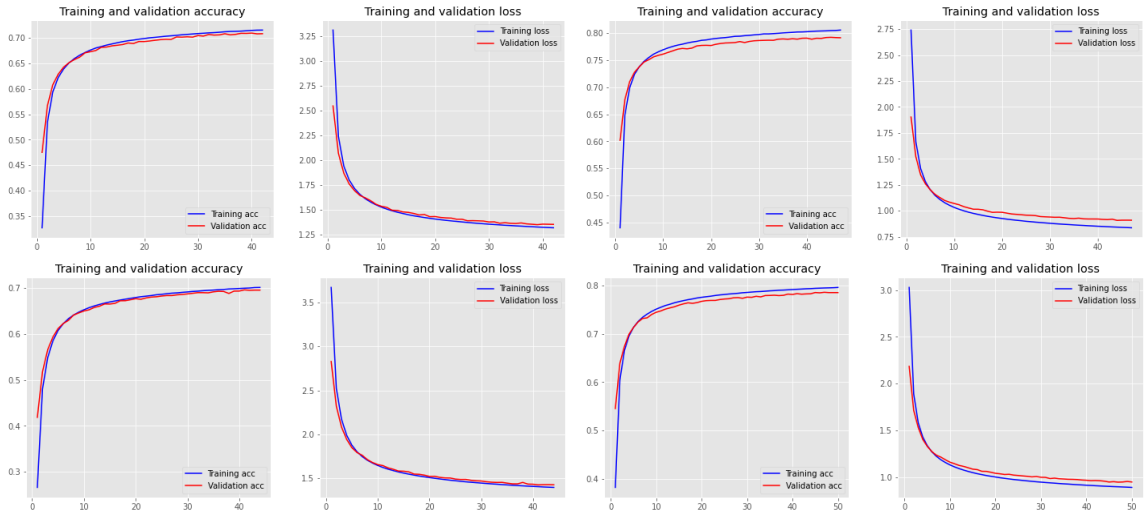
5.8. ábra. LSTM és BiLSTM (felül), GRU és BiGRU (alul)

A BiLSTM és BiGRU modellek teljesítményét tovább javítottam a batch_size módosításával, 128-as érték volt az optimális. Az így tanított hálózatok súlyait elmentettem.

5.0.2. Transfer Learning módszer alkalmazása

A következő kísérleteket a PFAM adathalmazon végeztem. Először a hagyományos módon tanítottam az előzőekben megismert alapmodelleket az adathalmaz segítségével. Az alapmodellt ugyanúgy konstruáltam, mint az UniProt adathalmaz esetében. A vizsgálatokat 32 és 64 neuronszámú modelleken végeztem el, nem használtam Dropout réteget. 270 darab, 5000 számosságú osztályon kezdtem a tanítást.

	BiLSTM	BiLSTM	BiGRU	BiGRU
Neurons	32	64	32	64
Train acc (%)	71.48	80.88	70.23	87.82
Val acc (%)	70.79	79.15	69.55	78.48
Test acc (%)	70.78	79.11	69.52	78.49

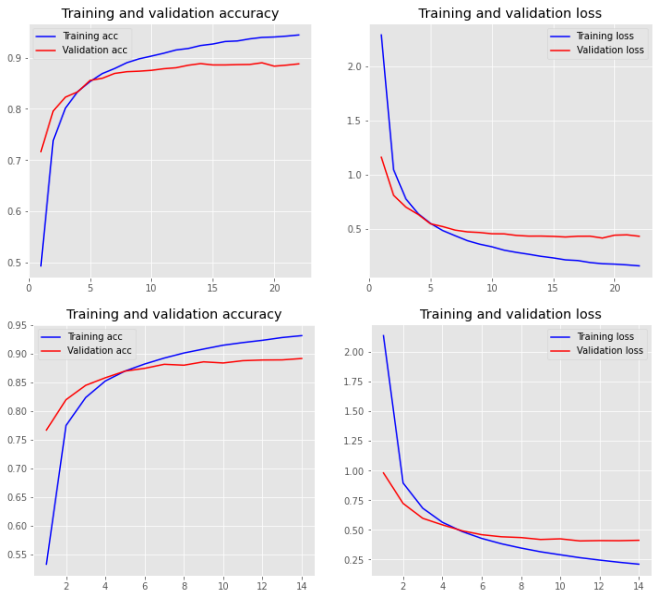


5.9. ábra. BiLSTM (felső sor) és BiGRU (alsó sor)

Az alapmodellek jó értékeket produkáltak, azonban nagyon lassú volt a tanulás folyamata, így nem tudtam érdemben kipróbálni a bonyolultabb felépítésű modelleket hosszabb szekvenciákkal. Próbálkoztam az osztályok darabszámának valamint számosságának csökkentésével. Azonban így sem értem el olyan eredményeket amiket vártam, ezért a transfer learning módszerét próbáltam meg alkalmazni. Az előzőleg elmentett modelleket felhasználva a tanítást beállított súlyokkal kezdtem.

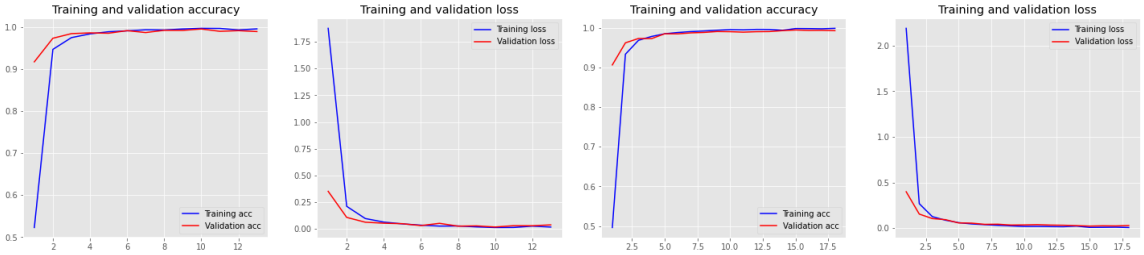
	BiLSTM	BiGRU
Neurons	128	128
Train acc (%)	96.48	95.21
Val acc (%)	88.78	89.21
Test acc (%)	89.1	88.85

A módszernek köszönhetően a 100 darab, 1000 számosságú osztályt, 300 hosszú szekvenciákkal tudtam tanítani a modellnek, és az RNN réteg neuronszámát is 128-ra tudtam növelni úgy, hogy a modell egyik kísérlet során sem tanult lassan. így a kísérlet sikeresnek mondható.

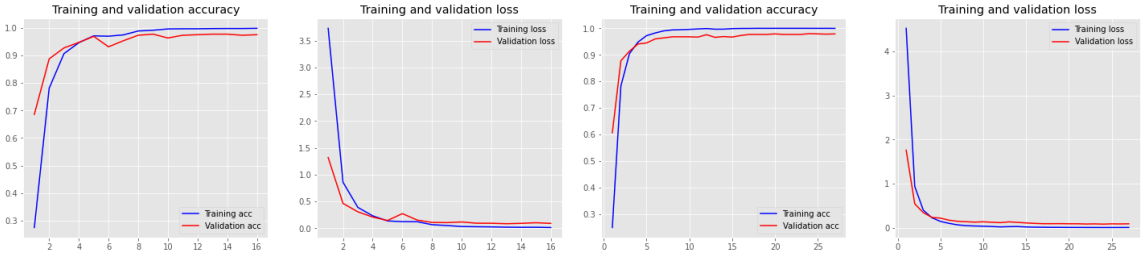


5.10. ábra. BiLSTM (felső sor) és BiGRU (alsó sor)

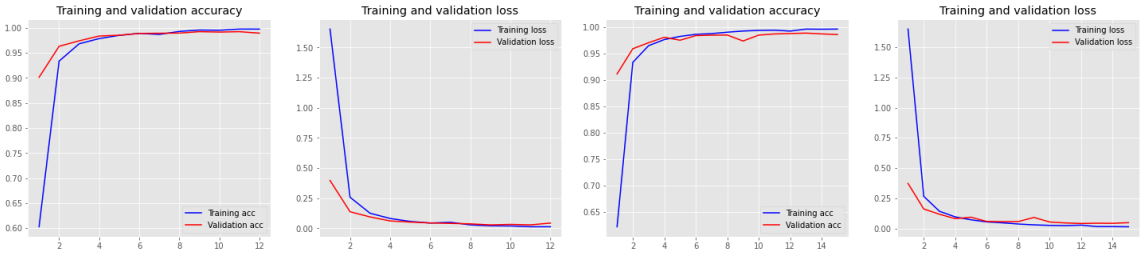
A Kaggle forrásból származó adatbázis szekvencikészlet szempontjából eltér a többitől, és az alapstatisztikák alapján kiegyenlítetlen az osztályok számossága. A modellt nagy számosságú osztályokon tanítottam, azt várva, hogy ezeken nagy pontosságot fog elérni. Majd a transfer learning módszer segítségével továbbtanítottam alacsony számosságú osztályokon. Négy előtanított modellel végeztem a kísérleteket. Egy-egy BiLSTM és GRU 50 darab, 1017 számosságú osztályon tanult, egy-egy pedig 100 darabb, 884 számosságú osztályon. A továbbtanitáshoz kiválasztottam 50 és 100 darab, 100 számosságú osztályt. Az eredmények a következő oldalon láthatóak. A kísérlet sikeresnek mondható, a módszer segítségével kiegyenlítetlen adathalmazt tudtam megtanítani a modellnek.



5.11. ábra. BiLSTM 50 és 100 darab osztályon tanítva.



5.12. ábra. BiLSTM 50 és 100 darab osztályon továbbtanítva.



5.13. ábra. BiGRU 50 és 100 darab osztályon tanítva.

5.14. ábra. BiGRU 50 és 100 darab osztályon továbbtanítva.

6. fejezet

Összefoglalás

A vizsgálatok során összességében a BiLSTM réteget alkalmazó hálózatok teljesítettek a legjobban, ebből arra lehet következtetni, hogy szekvenciális adatok esetében ilyen hálózatot érdemes alkalmazni. Emellett a BiGRU hálózatok is jól teljesítettek az adatokon, valamint az is elmondható, hogy a kétirányú rétegek jobb eredményeket értek el, mint a nem kétirányúak. A kísérletek az elvárásaimnak megfelelő eredményekkel zárultak, ugyanakkor fontos mejegyezni, hogy az adathalmaz tulajdonságai miatt értek el a modellek ekkora pontosságot. A jelenség azzal magyarázható, hogy a különböző osztályok határozottan elkülölnek egymástól, azaz a különböző osztályokba tartozó adatok távolsága nagy.

Irodalomjegyzék

[1] Michael Phi: "Illustrated Guide to LSTM's and GRU's: A step by step explanation" (2018) https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

[2] Christopher Olah: "Understanding LSTM Networks" (2015) https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[3] Martin T. Hagan, Howard B. Demuth, Mark Hudson Beale, Orlando De Jesús: "Neural Network Design" https://hagan.okstate.edu/NNDesign.pdf

[4] Horváth G. (szerk.), Altrichter M., Horváth G., Pataki B., Strausz Gy., Takács G., Valyon J.: "Neurális hálózatok", Budapest, Panem Kiadó, (2006)

[5] OpenStax: "Biology. Proteins" (2012) `https://cnx.org/contents/GFy_h8cu@9.85:2zzm1QG9@7/Proteins`

[6] IBM Cloud Education: "Recurrent Neural Networks" (2020) `https://www.ibm.com/cloud/learn/recurrent-neural-networks#toc-common-act-ydtr_7C6`

[7] Fei-Fei Li, Justin Johnson, Serena Yeung: "Lecture 10: Recurrent Neural Networks" (2017) `http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10`

[8] Sinno Jialin Pan, Qiang Yang: "A Survey on Transfer Learning" (2010) `http://www-edlab.cs.umass.edu/cs689/reading/transfer-learning.pdf`

[9] Joanne Quinn, Joanne J. McEachen, Michael Fullan, Mag Gardner, Max Drummy: "Dive Into Deep Learning: Tools for Engagement" (2019) `https://d2l.ai/chapter_preface/index.html#about-this-book`

[10] Ian Goodfellow, Yoshua Bengio, Aaron Courville: "Deep Learning" (2016) https://www.deeplearningbook.org/

[11] https://www.creative-biostructure.com/coronavirus/protein-sequence-analysis-p45.htm

[12] http://pfam.xfam.org/

[13] https://www.uniprot.org/

[14] https://www.kaggle.com/googleai/pfam-seed-random-split

NYILATKOZAT

Név: SZÖKRÖN DOROTYA

ELTE Természettudományi Kar, szak: MATEMATIKA

NEPTUN azonosító: DZHDN9

Szakdolgozat címe: GÉPI TANULÁS ALKALMAZÁSA MOLEKULÁRIS BIOLÓGIAI FELADATOKRA

A **szakdolgozat** 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. 31.

Szökrön Dorottya

a hallgató aláírása