Západočeská univerzita v Plzni

Fakulta Aplikovaných věd
KKY

Bakalářská práce

Automatické rozdělení DNA do funkčních oblastí

Autor:
Karel Paukner

Vedoucí práce:
Ing. Lukáš Kuhajda

2023

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Karel PAUKNER**
Osobní číslo:	**A19B0371P**
Studijní program:	**B0714A150005 Kybernetika a řídicí technika**
Specializace:	**Umělá inteligence a automatizace**
Téma práce:	**Automatické rozdělení DNA do funkčních oblastí**
Zadávající katedra:	**Katedra kybernetiky**

Zásady pro vypracování

I) Nastudování základní biologické terminologie potřebné pro vypracování úlohy.
II) Rešerše v oblasti využití umělé inteligence pro práci se sekvencemi DNA.
III) Vytvoření databáze DNA sekvencí, vytvoření trénovacího/testovacího datasetu.
IV) Návrh a natrénování modelu pro rozdělení DNA do množiny funknčních oblastí.
V) Analýza a vyhodnocení výstupů.

Rozsah bakalářské práce: **30-40 stránek A4**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná**

Seznam doporučené literatury:

Dodá vedoucí práce

Vedoucí bakalářské práce: **Ing. Lukáš Kuhajda**
Výzkumný program 1

Datum zadání bakalářské práce: **17. října 2022**
Termín odevzdání bakalářské práce: **22. května 2023**

Západočeská univerzita v Plzni
Fakulta aplikovaných věd
L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Prof. Ing. Josef Psutka, CSc.
vedoucí katedry

V Plzni dne 17. října 2022

Prohlášení

Předkládám tímto k posouzení a obhajobě bakalářskou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne 14. srpna 2023 _____

Poděkování

Chtěl bych upřímně poděkovat svému vedoucímu bakalářské práce, panu Ing. Lukášovi Kuhajdovi, za jeho neocenitelnou pomoc, podporu a cenné rady během celého procesu tvorby této práce. Vaše vedení, trpělivost a ochota byly pro mě klíčovými faktory při dosahování úspěchu v tomto projektu. Vaše odborné znalosti a rady mi pomohly lépe porozumět problému, který jsem zkoumal, a navrhnout efektivní řešení.

Abstrakt

Genomická analýza hraje klíčovou roli v pochopení biologických procesů a vývoje nemocí. V posledních letech se zaznamenal rapidní vzestup genomických dat, což vyžaduje efektivní metody pro rychlé a přesné rozpoznávání různých aspektů genomů. Tato práce se zaměřuje na využití transformeru pro rozpoznávání částí genomů. Dále výstupem práce bude jednoduchý skript na real-time vizualizaci predikce těchto částí.

Klíčová slova: biokybernetika, umělá inteligence, neuronové sítě, transformer, DNA, promotor

Abstract

Genomic analysis plays a pivotal role in understanding biological processes and disease development. In recent years, there has been a rapid surge in genomic data, necessitating efficient methods for swift and accurate recognition of various genomic aspects. This study focuses on harnessing a transformer for genomic segment recognition. Furthermore, the output of this study will be a simple script for real-time visualization of the prediction of these segments.

Key words: biocybernetics, artificial intelligence, neural networks, transformer, DNA, promoter

Obsah

Seznam obrázků

Úvod

Genom, genetická kniha života, nese informace, které určují nejen vzhled a charakteristiky organismů, ale i způsob, jakým se vyvíjejí a jak reagují na okolní prostředí. Otevření této knihy odhaluje genetické tajemství, jež může odhalit základy biologických procesů, vývoje nemocí a evolučních dějů. V posledních letech, díky pokročilým technikám analýzy a strojového učení, nabyly metody rozpoznávání částí genomu na důležitosti.

Cílem této práce je rozpoznávání jednotlivých částí genomu, což zahrnuje identifikaci klíčových elementů jako jsou promotorové oblasti, kódující oblasti a další. Každá z těchto částí nese významnou biologickou informaci, která přispívá k regulaci genové exprese, interakcím mezi buňkami a způsobům, jakými organismy reagují na vnější podněty.

Rozpoznání těchto částí genomu se však setkává s řadou výzev. Genom je komplexní a dynamický, jehož struktura může být ovlivněna různými faktory. K tomu, aby bylo možné správně identifikovat jednotlivé části genomu, je třeba kombinovat pokročilé techniky analýzy dat, strojového učení a bioinformatiky.

Práce bude rozdělena na 2 kapitoly. V první budou představeny důležité pojmy pro porozumění práce. V druhé kapitole už bude představeno samotné řešení problému klasifikace genomu pomocí neuronových sítí.

1 Důležité pojmy

V kapitole bude objasněno několik důležitých pojmů, na které bude v dalších částech práce odkázáno. První sekce bude obsahovat biologické pojmy. Dále zde budou představeny neuronové sítě a poslední sekce této kapitoly se zaměří na ukázku podobných prací.

1.1 Biologické pojmy

Bakalářská práce zahrnuje zpracování DNA sekvencí, proto zde budou čtenáři představeny biologické pojmy, které jsou nezbytné pro pochopení obsahu.

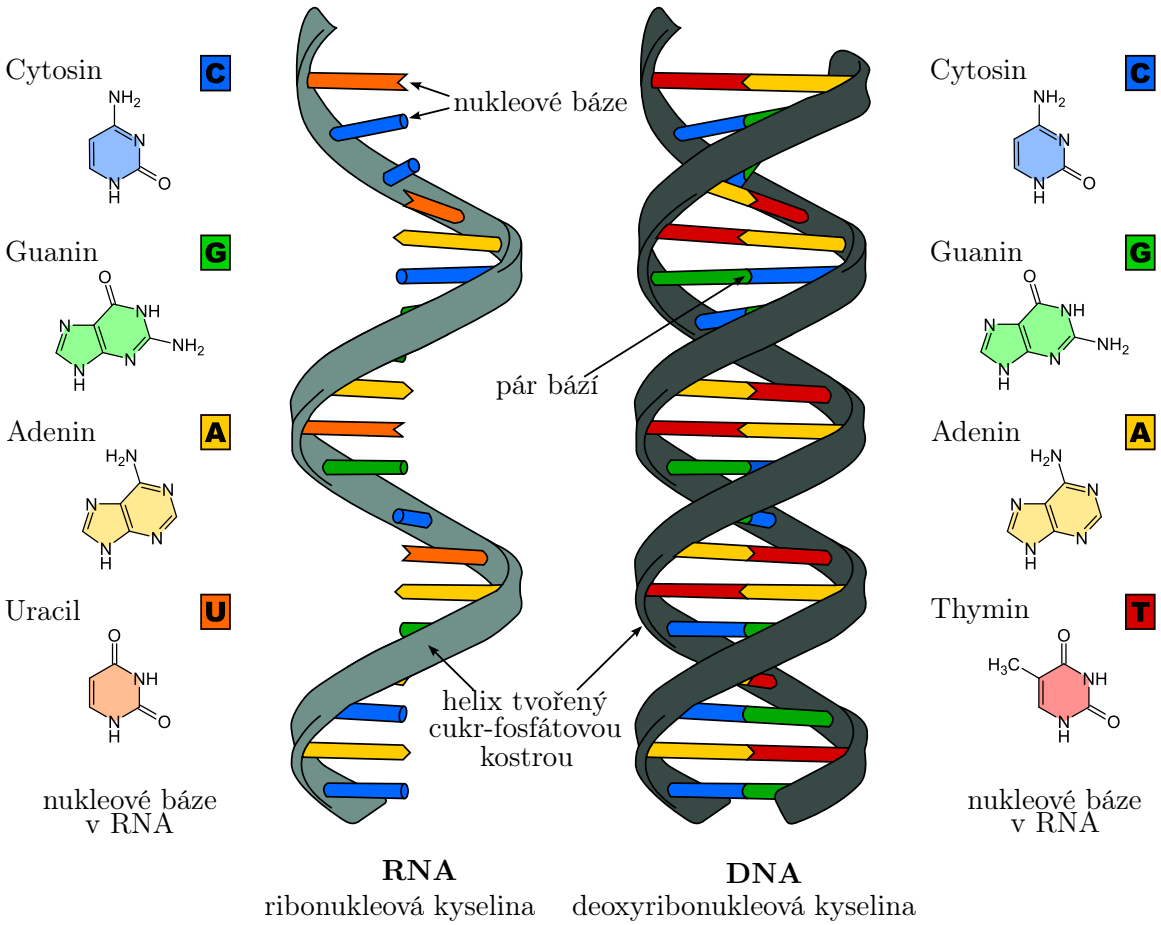
1.1.1 DNA

DNA neboli deoxyribonukleová kyselina je molekula nacházející se uvnitř buněk, která nese základ genetické informace živých organismů. Skládá se ze dvou kolem sebe vinoucích vláken, které připomínají zkroucený žebřík neboli dvoušroubovici. Každá šroubovice je složená z cukr-fosfátové kostry a čtyř základních nukleových bází: adeninu (A), cytosinu (C), guaninu (G) a thyminu (T). Každé vlákno může mít dva směry, které se určují podle propojení 5' a 3' uhlíku na začátku a konci kostry. Vlákno může tedy mít směr 3' -> 5' nebo 5' -> 3'. Každá dvoušroubovive se skládá ze dvou vláken, které mají opačný směr. [1]

Dvoušroubovice vzniká spojením nukleových bází. Toto spojení má svá pravidla, to taková, že se každá nukleová báze páruje jen s určitou další nukleovou bází:

- adenin (A) se vždy spáruje s thyminem (T)
- cytosin (C) se vždy spáruje s guaninem (G)

Pravidelné spárování bází nazýváme komplementarita a je viditelné na obrázku 1 . Díky Zmíněnému vztahu při zkoumání DNA stačí použít jen jedno vlákno (šroubovici). [2]



Obrázek 1: Porovnává DNA s RNA. Obrázek z [3]

1.1.2 Protein

Proteiny jsou nejuniverzálnější makromolekuly v živých systémech. Plní mnoho klíčových funkcí v podstatě ve všech biologických procesech. Fungují jako katalyzátory, transportují a ukládají další molekuly jako je kyslík, poskytují imunitní ochranu, vytvářejí pohyb, přenášejí nervové vzruchy a řídí růst. Proces vzniku proteinu dělíme na dvě části. Tyto části se nazývají Transkripce a Translace. Obě tyto části budou představeny v další podsekci. [4]

1.1.3 Geny

Geny jsou základní jednotky dědičnosti a genetické informace v živých organismech. Určují charakteristiky, jako barvu očí, typ krevní skupiny, strukturu enzymů, hormony a mnoho dalších vlastností. Při reprodukci se geny předávají z jednoho rodiče na potomstvo a tím zajišťují přenos dědičných vlastností a vývoj nových generací živých organismů. Navíc geny ovlivňují reakci organismu na prostředí a mohou být také zapojeny do rozvoje onemocnění. Jsou to úseky DNA, které obsahují kódy (sekvence nukleotidů) nezbytné pro vytváření specifických proteinů. [5]

Gen se skládá ze 3 hlavní částí a to z promotoru, kódovací sekvence (Coding sequence (CDS)) a terminátoru.

1.1.3.1 Promotor

Promotor je klíčovou regulární oblastí na molekule DNA, která se nachází hned před CDS. Obsahuje specifické sekvence nukleotidů, které umožňují vázání transkripčních faktorů, které ovlivňují aktivitu RNA polymerázy, což je enzym, který provádí samotnou transkripci.

Transkripce začíná na místě zvaném jako "startovací místo transkripce (transcription start site(TSS)). Proces transkripce neboli přepisu si můžeme představit jako rozdělení DNA dvoušroubovice v místě genu a přepsání jednotlivých bází z jedné šroubovice na novou ve směru 5' -> 3', kterou nazýváme mRNA (Messenger RNA) a je zobrazena na obrázku 1. Tyto báze se přepisují hlavně z druhé části genu CDS. Transkripce končí terminátorem. Jediný rozdíl při přepisu je, že nukloevá báze Thyimin (T) je přepsána jako Uracil (U). [6]

U promotoru se udává, že by měl být tvořen 100 - 1000 nukleovými bázemi. Dále se udává, že by měl blíže k CDS obsahovat sekvenci bází ve tvaru TATA. Tedy čím blíže k CDS, tím by mělo být snazší promotor rozpoznat. [7]

1.1.3.2 CDS

CDS je úsek DNA nebo RNA, který obsahuje genetickou informaci pro syntézu proteinů. Tato sekvence obsahuje tripletové kódy, nazývané kodony, které představují jednotlivé aminokyseliny, ze kterých jsou proteiny složeny. Během procesu translace je CDS přeložena pomocí ribozomu do řetězce aminokyselin, čímž se vytváří protein s konkrétní sekvencí aminokyselin, který má specifickou funkci v buňce nebo organismu. CDS je tedy klíčovou částí genetické informace, která určuje strukturu a funkci proteinů. Translace začíná kodonem s podobou AUG (Adenin, Uracil, Guanin) a končí kodony UAA, UAG nebo UGA. [4]

Díky vlastnosti translace má CDS velice jasnou strukturu, která musí obsahovat jednotlivé kodony pro překlad do aminokyselin, tedy u následného rozpoznání této části genomu by neměl být problém.

1.1.4 Offgene

Offgene je třetí a poslední část genomu, která bude v práci rozpoznávána. Tato část je pro naši práci určena jako vše co nepatří do CDS nebo promotoru. Díky tomu by měla být hůře rozpoznávána než ostatní dvě části.

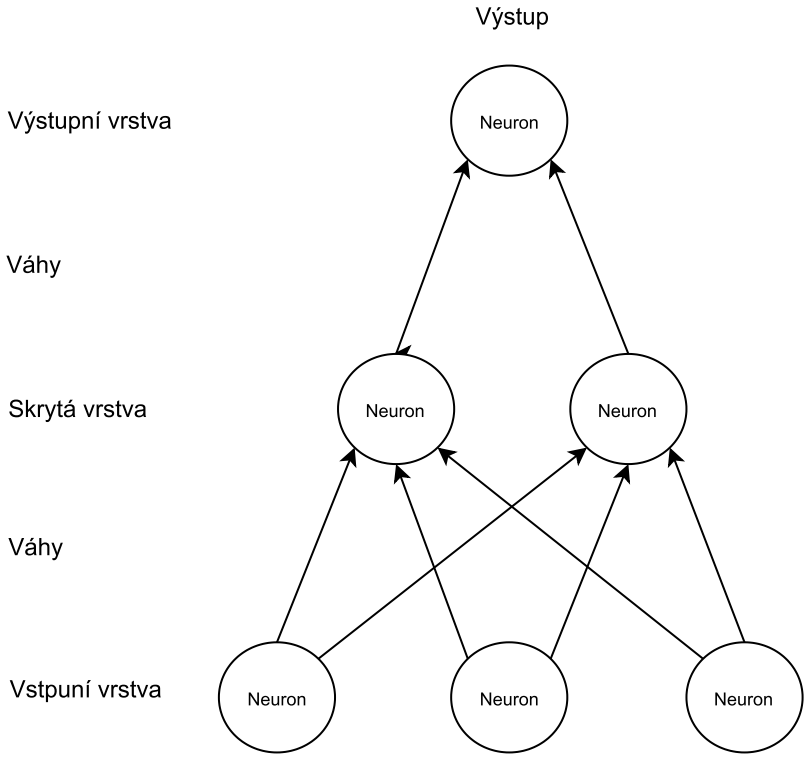
1.2 Neuronové sítě

V této sekci budou nejprve vysvětleny základy učení neuronových sítí a poté samotné architektury, které se v práci objeví v dalších částech.

Neuronové sítě jsou matematickým nelineárním modelem inspirovaným biologickou strukturou mozku, který slouží k řešení složitých úloh. Tyto sítě jsou založeny na spojení jednotlivých prvků nazývaných neurony, umožňujících simulovat určitou úroveň inteligence na našich počítačových strojích. [8]

1.2.1 Učení neuronových sítí

Vysvětlení učení neuronových sítí zde bude předvedeno na architektuře známé jako více vrstvá perceptronová síť. Více vrstvé perceptronové síťě se skládají z několika vrstev neuronů. První vrstva této sítě je vždy vstupní vrstva neuronů poté následuje jedna nebo více skrytých vrstev neuronů a na konci se nachází výstupní vrstva neuronů. Každé spojení mezi neurony je váženo číslem, nazývaným váha. Tato architektura je zobrazena na obrázku 2. Výstup jednotlivých neuronů ve skrytých vrstvách je určen aktivační funkcí, která přidává do sítě nelinearitu a zabraňuje paralýze sítě divergentními neurony.



Obrázek 2: Architektura perceptronové neuronové sítě

Výpočet výstupu každého neuronu může vypadat například takto:

$$o = \sigma(\sum_{j=1}^{N} V_j x_j + T_i^{hid}) \tag{1}$$

kde $\sigma()$ je aktivační funkce, N je počet neuronů v předešlé vrstvě, V_j jsou váhy, x_j jsou jednotlivé vstupy a T_i je práh (bias) vrstvy kde se daný neuron nachází. Příkladem aktivační funkce může

například být sigmoid funkce:

$$\sigma(u) = \frac{1}{1 + \exp(-u)} \tag{2}$$

Učení neuronové sítě probíhá postupnou aktualizací jednotlivých vah, které spojují jednotlivé neurony, tak aby síť byla schopná vykonávat určité úkoly, pro které byla vytvořena . Na začátku učení mají tyto váhy náhodné hodnoty. Aby se model dostal do stavu kdy může aktualizovat váhy, musí v něm nejdříve proběhnout takzvaná dopředná propagace (forward propagation).

1.2.1.1 Forward propagation

Metoda Forward propagation vezme data a nechá je projít jednotlivými vrstvami modelu. Postupným průchodem se vypočítají jednotlivé výstupy každé vrstvy (výpočet zde 1). Na výstupu sítě se poté pomocí ztrátové funkce (loss function) spočítá chyba.

1.2.1.2 Ztrátová funkce a výpočet chyby

Ztrátová funkce (loss function) je důležitá funkce sloužící k výpočtu chyby mezi požadovanou hodnotou výstupu určenou vstupem a výstupem neuronové sítě. Výpočet chyby může vypadat například takto:

$$E = \frac{1}{N} \sum_{i} (o_i - t_i)^2 \tag{3}$$

kde N je počet výstupních neuronů, o je výstup jednotlivých i neuronů výstupní vrstvy a t je i-tý požadovaný výstup. Poté nastává aktualizace neboli učení jednotlivých vah pomocí metody zpětné propagace (backpropagation). Zpětná propagace bude vysvětlena v její vlastní podsekci.

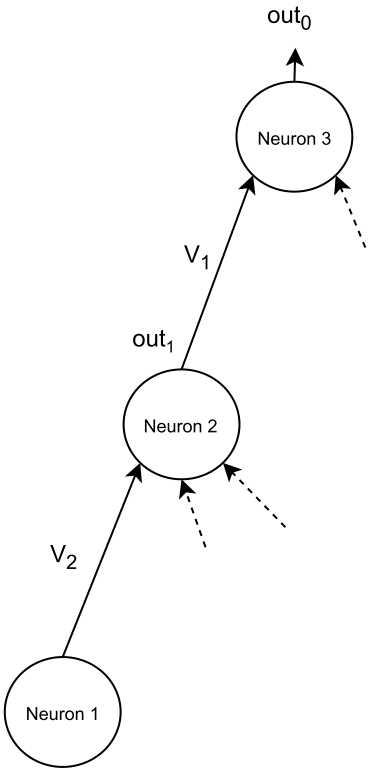
Ztrátová funkce použitá zde (3) se nazývá střední kvadratická chyba. Existuje mnoho dalších ztrátových funkcí. V této práci bude například použita křížová entropie:

$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{4}$$

Kde M je počet tříd do kterých se klasifikuje, y je binární indikátor, který říká jestli pro výsledek o je správný label c a p je předpokládaná pravděpodobnost, že výsledek o patří do třídy c

1.2.1.3 Zpětná propagace (backpropagation)

Metoda zpětné propagace je optimalizační proces, který se zaobírá minimalizací ztrátové funkce a tedy navýšením přesnosti modelu pomocí aktualizací jednotlivých vah. Pro dosažení těchto cílů se používá algoritmus gradientního sestupu, který pomocí derivací hledá lokální minimum ztrátové funkce. Zisk nové konfigurace vah probíhá od výstupní vrstvy až ke vstupní (proto zpětná propagace). Pro ukázku algoritmu zde bude výpočet předveden na části perceptronovém modelu, který byl definován výše (2) :



Obrázek 3: Část architektury perceptronové neuronové sítě

Jak je již psáno výše zpětná propagace probíhá od výstupní vrstvy ke vstupní, proto zde bude výpočet prezentován nejdříve na váze V_1 a až poté na V_2. Váhy jsou zobrazeny na obrázku 3:

$$V_{new} = V_{old} + \eta \Delta V \tag{5}$$

Tento vzorec lze použít pro obě váhy. Značí jak vypočítat novou hodnotu váhy pomocí staré hodnoty váhy, η parametru učení (learning rate) neuronové sítě, což je měřítko, které určuje, jak rychle se váhy modelu upravují v závislosti na gradientu ztrátové funkce a změně váhy (gradientu) ΔV .

Změna ΔV pro váhu V_1 se vypočte například takto :

$$\Delta V_1 = -\frac{\partial E}{\partial o} \frac{\partial out_0}{\partial V1} \tag{6}$$

kde E je chyba (výpočet zde 3) a out_0 je výstup z neuronu 3. Výpočet ΔV_2:

$$\Delta V_2 = -\frac{\partial E}{\partial o}\frac{\partial out_0}{\partial out_1}\frac{\partial out_1}{\partial V2} \tag{7}$$

Pokud by zde byla další vrstva výpočet by vypadal obdobně jen by se rozšířil o další derivaci. Tento jev je známy pod jménem řetízkové pravidlo.

Takže hlavní idea gradientního sestupu je získat ztrátovou funkci → vypočítat gradient → aktualizovat všechny váhy neuronové sítě. Toto opakovat dokud nedojde ke konvergenci. [9]

1.2.1.4 parametr učení neuronové sítě

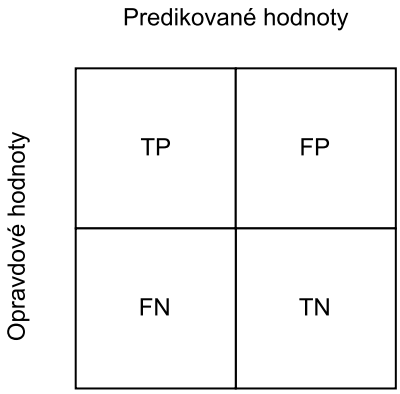
1.2.1.5 Metriky pro ohodnocení kvality modelu

Metriky používané k hodnocení modelu neuronových sítí (NN) jsou klíčové pro zhodnocení jejich kvalit. Mezi tyto metriky patří hodnota ztrátové funkce, která již byla popsána výše 1.2.1.1 a přesnost neboli accuracy. Přesnost udává podíl korektně klasifikovaných vzorků na celkovém počtu vzorků. Výpočet může vypadat například takto:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

kde TP (True Positive) je počet správně identifikovaných vzorků, TN (True Negative) je počet správně identifikovaných negativních vzorků a dále FP (False Positive) a FN (False Negative) jsou nesprávně identifikované pozitivní a negativní vzorky.

Další metrika využívaná pro ohodnocení kvality modelu je matice zmatení (confusion matrix). Slouží k vizuální reprezentaci výkonu klasifikačního modelu, která pomáhá analyzovat a kvantifikovat jeho předpovědi a chyby.



Obrázek 4: Matice zmatení

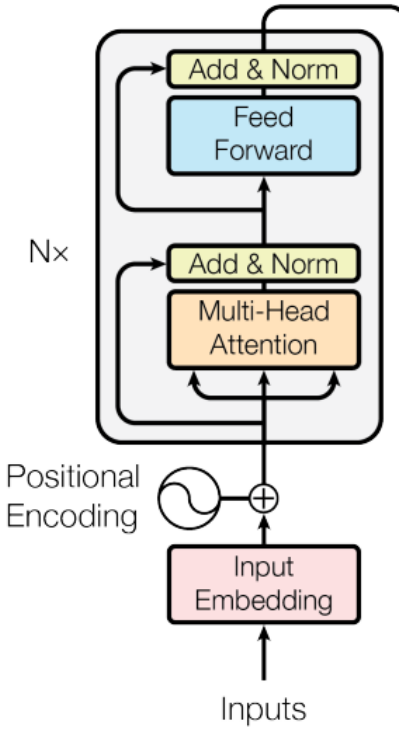
1.2.2 Architektury neuronových sítí

V této sekci budou ukázány jednotlivé architektury neuronových sítí. Nejdříve zde budou uvedeny architektury, které budou použity v praktické části jako například transformer a konvoluční neuronová síť, dále budou architektury, které se objeví v sekci 1.7

1.2.2.1 Transformery

Transformery jsou revoluční modely neuronové sítě, které změnily způsob, jakým zpracováváme přirozený jazyk. Transformery přispěly k úspěchům v oblastech jako strojový překlad, generování textu, analýza sentimentu a mnoho dalších. Prvně byli představeny v roce 2017 ve článku "Attention is All You Need"[10]. Základním mechanismem tranformerů je právě attention (pozornostní) vrstva, která jim umožňuje se zaměřit na některé části vstupního textu.

Transformer může být vytvořen ze dvou pod-částí, a to enkodérem (encoder) a dekodérem (decoder). Slovo "může"je zde uvedeno, protože pro některé úlohy stačí použít jen Enkodér nebo samotný Dekodér. Například v této práci bude poté pro klasifikaci využit jen enkodér.



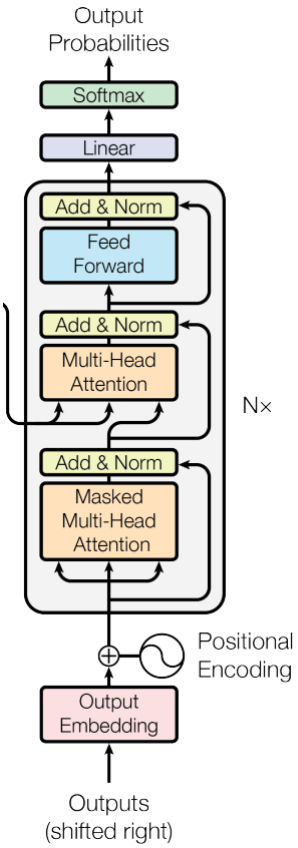
Obrázek 5: Architektura enkodéru. Obrázek z [10]

obě části se skládají z transformer bloků, které se skládají z:

– Multi-Head Self-Atention vrsta (více-hlavová pozornostní vrstva) - Tato vrstva umožňuje modelu získat informace o vzájemných závislostech mezi slovy. Více takzvaných hlav slouží

k nalezení více závislostí, což vede k lepšímu pochopení kontextu.

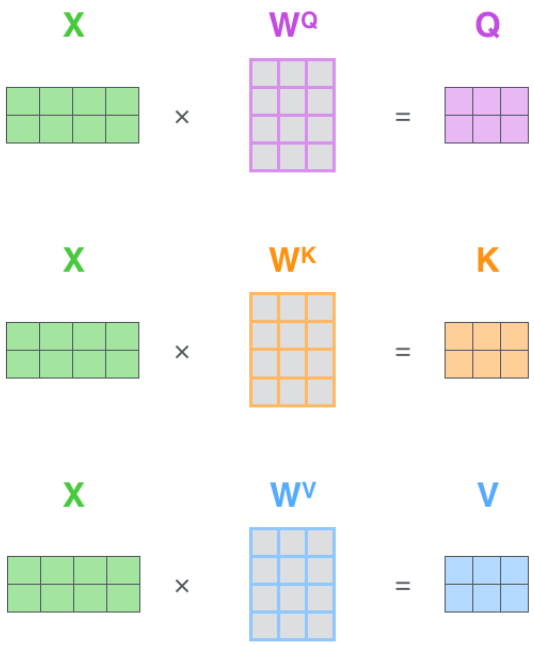
– Feed-Forward Neural Networks - Jednoduché neuronové sítě, které dostávají jako vstup výsledek z Multi-Head Self-Atention vrstvy a jejich úkolem je zpracovat výstup z jedné Atention vrstvy a připravit vstup pro falší transformer blok.



Obrázek 6: Architektura dekodéru. Obrázek z [10]

Multi-Head Self-Atention vrstva

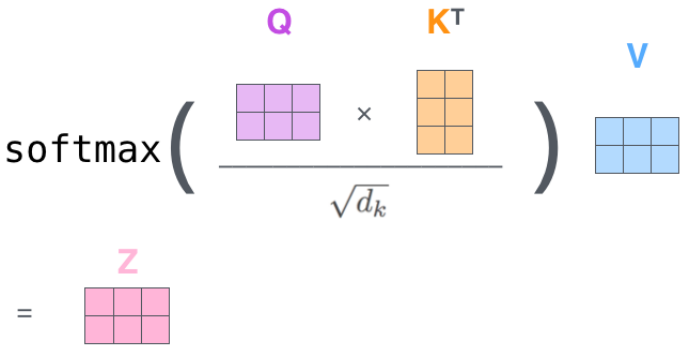
Pro výpočet takzvané self-attention je zapotřebí vypočítat tři matice a to Query dále jen jako Q, Key dále jen jako K a Value dále jen jako V. Tyto matice se získají vynásobením vstupu se třemi odlišnými maticemi, které se v průběhu trénují. Tyto matice se označují jako W^Q , W^K a W^R. Výpočet je znázorněný na obrázku 7.



Obrázek 7: Výpočet matic Q, V, a K. Obrázek převzat z [11]

Matice x je vstup (matice čísel), která se získá pomocí součtu embedingu s positional encodingem. Každá řádka této matice patří jednomu slovu ze vstupní věty. Positional encoding zde zajišťuje schopnost modelu rozlišovat význam slov na základě jejich umístění ve větě. [12]

Jak vypočítat výslednou attention matici je znázorněno na obrázku 8. d_k odpovídá dimenzi matice K. Dělení odmocninou je zde z důvodu více stabilního gradientu.



Obrázek 8: Výpočet výsledné attention matice. Obrázek převzat z [11]

1.2.3 Konvoluční sítě

Konvoluční síť neboli convolution neural network (CNN) [13] je specializovaná neuronová síť určená pro zpracování vizuálních dat, jako jsou obrázky a videa, ale funguje také dobře pro neobrazová data. CNN, jak napovídá z názvu, využívá operace konvoluce, což umožňuje efektivní zpracování vizuálních informací. Dále je už běžným zvykem dávat za výstup CNN takzvanou

pooling vrstvu, která bude tedy představena v této sekci též.

V praktickém světe pythonu existují různé typy implementací CNN a ty dělíme na 3d, 2d a 1d implementace. 2d CNN jsou asi nejvyužívanější, hlavně proto, že se používají v oblasti rozpoznávání obrázku [14] (obrázek odpovídá matici o dimenzi 2), což není cíl této práce. Proto zde bude představena 1d CNN.

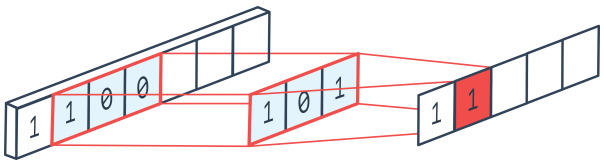
1d CNN síť funguje tak, že konvoluční jádro, nazývané jako kernel, se postupně posouvá přes 1 dimenzionální data a vypočítává hodnotu konvoluce v každém bodě. Toto zde bude předvedeno na menší ukázce:

- Mějme data x o délce n, kernel h o délce k a kernel se posouvá po datech o hodnotu s. Výpočet nekauzální konvoluce vypadá takto:

$$y(n) = \begin{cases} \sum_{i=0}^{k} x(n+i)h(i), & n = 0 \\ \sum_{i=0}^{k} x(n+i+(s-1))h(i), & n \neq 0 \end{cases} \tag{9}$$

- Dále mějme parametry dat a konvoluce nadefinovány takto $n = 5$, $k = 3$ a $s = 1$. Výstup konvoluce bude poté vypadat tímto způsobem:

$$y(0) = x(0)h(0) + x(1)h(1)x(2)h(2)$$
$$y(0) = x(0)h(0) + x(1)h(1)x(2)h(2)$$
$$y(0) = x(0)h(0) + x(1)h(1)x(2)h(2)$$



Obrázek 9: Ukázka konvoluce. Obrázek z [15]

Dále je zapotřebí uvést takzvaný padding, který na konce a začátky vstupních dat přidá 0. Tento parametr pak ovlivňuje velikost výstupu. Například výše je vidět, že výstup je menší než vstup a to proto, že zde nebyl použit žádný padding. V pythonu máme různé možnosti paddingu, zde si ukážeme jen 2 základní:

- "Valid", který znamená žádný padding, viz příklad výše.
- "Same"přidá tolik nul na kraje, aby výstup byl stejně dlouhý jako vstup. Tento padding bude použit později.

1.2.3.1 Pooling

Jako u konvoluce, i zde platí, že máme různé implementace poolingu. Pooling má velice podobné parametry jako konvoluce. Důležitým parametrem je velikost pooling okna, hodnota posunu tohoto okna a padding. Padding je zde naprosto stejný jako u konvoluce (same a valid).

Z důvodu jednoduchosti ukázky 1d poolingu přesněji Max-poolingu zde bude i předveden: (další známá funkce poolingu je například Max-pooling, který funguje velice podobně jen místo maximální hodnoty v okně vrací průměrnou). [16]

Mějme nějaký vektor dat, například:

$$data = [1, 2, 3, 4, 5]$$

Nastavíme si 1d Max-pooling s těmito parametry:

$$velikost_{okna}(pool - size) = 2$$
$$Padding = "valid"$$
$$Posun_{okna}(strides) = 1$$

Pokud projedeme našimi daty přes tuto pooling vrstvu dostaneme:

$$vysledek = [2, 3, 4, 5]$$

Zde je opět vidět, že velikost výstupních dat je menší než vstupních a to kvůli nastavenému valid paddingu. Pokud by se změnil parametr paddingu na same, tak by výsledek vypadal takto:

$$vysledek = [2, 3, 4, 5, 5]$$

1.3 Dense layer

Dense layer, známá také jako hustá vrstva nebo plně propojená vrstva, je klíčovým stavebním blokem neuronových sítí ve strojovém učení. Tato vrstva hraje důležitou roli při zpracování dat a učení vzorů. Každý prvek ve vstupním vektoru je propojen se všemi prvky ve výstupním vektoru váhami, což umožňuje modelu hledat komplexní vztahy mezi daty. Umožňuje vykonávat úkoly, jako je klasifikace nebo regrese, na různých typech dat. [17]

1.4 Droput layer

Dropout vrstva je technika používaná v k omezení přetrénování (overfittingu) a zlepšení generalizace modelu. Tato technika byla poprvé představena v práci "Dropout: A Simple Way to Prevent Neural Networks from Overfitting"od Geoffreyho Hintona a jeho kolegů. [18]

Idea dropout vrstvy spočívá v tom, že během tréninku se náhodně deaktivují (vypnou) určité jednotky nebo neurony v síti. Tímto způsobem se síla závislosti mezi neurony snižuje, a tím dochází k "vypnutí"určitých vazeb v síti. To pomáhá modelu zabránit takzvanému přílišnému naučení na konkrétních vzorcích ve tréninkových datech a podporuje generalizaci na nová, dosud neviděná data.w

1.4.1 Overfitting

Přetrénování (Overfitting) je základní problém ve strojovém učení, který brání dokonale generalizovat modely tak, aby dobře odpovídaly pozorovaným datům na tréninkových datech, stejně jako na nových datech. Kvůli přítomnosti šumu, omezenému počtu tréninkových vzorků a složitosti klasifikátorů dochází k přetrénování. [19]

1.5 Rekurentní neuronové sítě

U rekurentních neuronových sítí se ke známé architektuře neuronové sítě z obrázku 2 přidá ještě takzvaná zpětná vazba, která neuronové síti přidá takzvanou 'paměť' neboli stav.

Výstup Neuronu i ve skryté vrstvě se zde rovná:

$$h_i(t) = \sigma(\sum_{j=1}^{N} V_{ij} x_j(t) + \sum_{k=1}^{H} W_{ik} h_k(t-1) + T_i^{hid}) \tag{10}$$

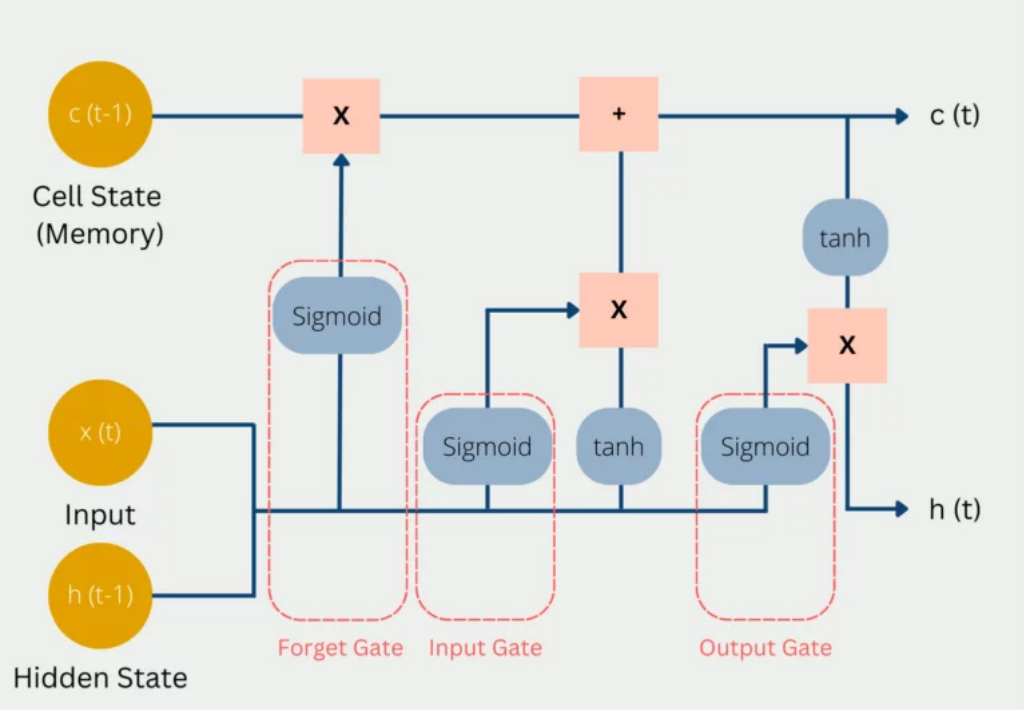
Kde $h_k(t-1)$ je nový vstup, který je zkopírován ze skryté vrstvy v čase (t-1) a W_{ik} jsou váhy mezi novým vstupem a skrytou vrstvou.

Z důvodu problémů rekurentních neuronových sítí (RNN), které nemohly efektivně zpracovávat dlouhá časová zpoždění kvůli problémům s tokem chybových signálů, protože chyba rostla nekonečně a nebo exponenciálně klesala vznikla architektura Long Short Term Memory (LSTM).

Jako RNN tak i LSTM má takzvanou paměť, která funguje pomocí speciálních bloků, které jsou vzájemně propojené a slouží jako paměťové jednotky. Tyto bloky fungují jako diferencovatelná verze paměťových čipů v počítači. Každý blok obsahuje paměťové buňky, které mohou uchovávat informace, a tři brány - vstupní, výstupní a zapomínací. Tyto brány umožňují kontrolovaně zapisovat, číst a resetovat hodnoty v paměťových buňkách. Interakce sítě s těmito buňkami

probíhá prostřednictvím brán. Výhodou LSTM je schopnost zpracovávat dlouhodobé závislosti v datech, což se osvědčilo při řešení reálných problémů, jako je rozpoznávání izolovaných slov a kontinuální rozpoznávání řeči. [20]

V dnešní době se už místo LSTM používají více transformery.



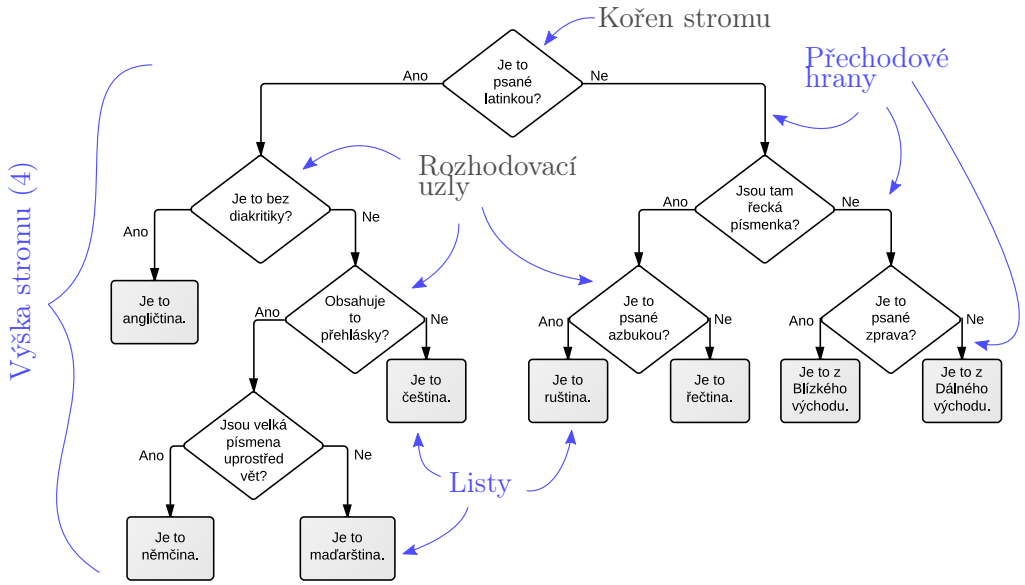
Obrázek 10: Architektura LSTM. obrázek z [21]

1.6 Random forest

Random forest [22] neboli náhodný les se používá pro klasifikaci. Skládá se z mnoha rozhodovacích stromů (decision trees), proto se nejdříve podíváme na ně.

Rozhodovací strom je tvořen z kořene stromu, přechodových hran, rozhodovacích uzlů a listů. Použití rozhodovacího stromu je velice jednoduchá činnost. Kořen stromu a rozhodovací listy obsahují jednotlivé otázky, které jsou důležité pro následnou klasifikaci. Přechodové hrany odpovídají na jednotlivé otázky odpovědí ano/ne a pokračují dál k další otázce a nebo listu. Listy jsou už jednotlivé klasifikační třídy. Na obrázku 11 můžeme vidět jednoduchý příklad takového stromu.

Kvůli lepší přesnosti se místo samotného rozhodovacího stromu používá více takovýchto stromů dohromady. Každý strom se trénuje na jiné podmnožině dat. Podmnožina dat je vždy náhodně vybrána, proto náhodný les. Výsledek Random forestu je získán takzvaným hlasováním všech stromů, které se ve forestu nachází, neboli třída, která získala nejvíce hlasů je výstupem.



Obrázek 11: Příklad rozhodovacího stromu. Obrázek z [23]

1.7 Použití AI na rozpoznání částí genomu z jiné odborné práce

V této sekci budou uvedeny některé práce a jejich výsledky, které se zaobírali klasifikací genomu, přesněji binární klasifikací promotor/ne-promotor.

1.7.1 Comparison of machine learning and deep learning techniques in promoter prediction across diverse species

Ve článku z roku 2021 [24] se autoři zaobírají problémem rozpoznání promotoru pomocí neuronových sítí.

Autoři zde použili 3 genomy, a to A. thaliana genom, genom kvasnice a lidský genom. Tyto genomy byly vybrány pomocí UCSC genome browser [25]. Ze všech tří získali okolo něco přes 100000 sekvencí. Následně po menší úpravě dat vybrali náhodně 76000 sekvencí.

Z těchto sekvencí následně vyextrahovali promotory. Každý promotor byl vybrán okolo známého TSS (startovací místo transkripce) a to 700 bází před a 300 bází po TSS, tedy dohromady každý promotor byl o délce 1000 bází.

Tyto data byly následně rozděleny na dvě skupiny. První trénovací skupina obsahovala 90% dat a druhá testovací zbylých 10% dat. Dále byly data upraveny pomocí procesu "k-merizace". Tento proces rozdělí sekvenci na několik takzvaných "slov"(k-merů). K-merizace má dva důležité parametry:

1. Velikost K-meru: značí velikost "slova"
2. Posun (strides): značí posunutí (od jaké další báze se má vytvořit další k-mer)

Dále na data upravená k-merizací byly použity dva endcoding (kódovací) procesy:

1. frekvenční tokenizace (frequency-based tokenization): každý k-mer v sekvenci zakódován na základě jeho frekvence výskytu a přiřazen k jedinečnému indexu, začínajícím od 1.
2. one-hot encoding: každý k-mer reprezentován novou binární hodnotou.

Pro porovnání zde autoři těmito daty následně natrénovali CNN, LSTM a RF.

1.7.1.1 Výsledky

První výsledky ukazují vliv kódovacích technik na trénovaní CNN. Z tabulky 1 je vidět, že frekvenční tokenizace dosahovala lepších hodnot, jak časových tak i přesnostních.

Dále byly porovnávány jednotlivé modely mezi sebou v binární klasifikaci (je promotor / není promotor). Výsledky z druhé tabulky naznačují, že pro analýzu DNA sekvencí má CNN výhodu v porovnání s LSTM a RF. CNN dosahuje stabilně vysoké přesnosti nezávisle na velikosti k-meru, což je výhodné z hlediska praktického využití.

1.7.2 PromoterLCNN: A Light CNN-Based Promoter Prediction and Classification Model

Jak napovídá z názvu, článek se zaměřuje na podobný problém jako ten předchozí, tedy klasifikaci promotor/ne-promotor. Navíc je zde takzvaná druhá fáze klasifikace, kdy se jednotlivé promotory rozdělují do 6 σ tříd u bakterie zvané Escherichia coli. [26]

Z této bakterie je vybráno 2860 sekvencí promotoru a 2860 sekvencí ne-promotoru pro trénování. Pro testování 256 promotoru a 0 ne-promotru. Rozdělení těchto sekvencí promotoru do jednotlivých σ vrstev je možné vidět v této tabulce.

Jako modely byly zvoleny 2 CNN, které jsou poskládány za sebou. První pro rozpoznání samotné promotor sekvence a druhá pro klasifikování do jednotlivých σ tříd.

Parametry těchto sítí byly vybrány pomocí Hyperparameter Tuningu [27].

1.7.2.1 Výsledky

Výsledná přesnost prvního CNN modelu je 86%. Vážený průměr přesností druhého modelu s ohledem na počet prvků v každé třídě je 96,7 %.

2 Vlastní přínos

Kapitola bude rozdělena na dvě části. První část bude obsahovat popis získání datasetu sloužícího pro trénování. Druhá část následně předvede samotný model.

2.1 Dataset

2.1.1 Stažení dat

Data potřebná pro tvorbu datasetu byla stažena pomocí python skriptu ze stránky NCBI (National Center for Biotechnology Information), což je vládní a veřejná organizace Spojených států amerických, která se specializuje na poskytování informací a zdrojů v oblasti biologického a bioinformatického výzkumu.

Z této stránky byly staženy sekvence DNA 491 organismů hub. Tyto data byly vybrána, protože by měla být jednoduší pro klasifikaci než například lidská DNA. Pro následnou extrakci bylo nutné stáhnout od každého organismu 2 soubory.

1. .GFF soubor - Formát souboru používaný k popisu struktury genetických a genomických funkcí v biologických sekvencích
2. .fna soubor - Formát souboru obsahující textové zápisy sekvencí nukleových kyselin, kde jednotlivé záznamy zahrnují záhlaví (header) a řádky obsahující nukleotidové zápisy samotných sekvencí. Záhlaví může například obsahovat informace o identifikátoru sekvence, názvu a popisu.

Dohromady všechny tyto soubory obsahovali 124612 sekvencí genomu o průměrné délce 128662,89 bází. Poté byly sekvence vyextrahovány a rozděleny na CDS, Promotor a Offgene.

2.1.2 Parsování dat

Tato pod-sekce bude obsahovat popis extrakce jednotlivých částí genomu ze souboru fna za pomoci souboru gff. Extrakce je rozdělena na dvě části.

2.1.2.1 Zpracování GFF souboru

V tomto souboru byly nejdříve hledány řádky, které obsahují popis začátku a konce CDS, které byly uloženy do listu s informací směru dáne CDS sekvence (- nebo +). Následně bylo od začátku

nebo konce každé CDS sekvence (záleží podle směru) vybráno 1000/500/300 nukleových bází, které byly označeny jako promotor. Promotor byl ještě uložen s informací jak daleko se nachází další CDS. Poté zbylé nukleové báze, které nebyly označeny jako promotor nebo CDS byly klasifikovány jako Offgene.

2.1.2.2 Extrakce z FNA

Pomocí informací z GFF souboru (začátek sekvence, konec sekvence) byly vyextrahovány samotné nukleové báze a uloženy do nových fna souborů. U promotorů a CDS byl ještě před uložením u sekvencí s opačným směrem (-) provedeno otočení sekvence a převedení na reversní báze.

Pro lepší porozumění otočení a převedení na reversní bázi zde bude uveden příklad:

Mějme například sekvenci s těmito bázemi:

$$seq = "ATCG"$$

Nejdříve provedeme otočení (jestli nejdříve bude provedeno otočení nebo převedení je jedno):

$$otocena\_seq = "GCTA"$$

Poté na otočenou sekvenci provedeme převedení. Převedení probíhá vždy na komplementární bázi.

$$vysledna\_seq = "CGAT"$$

Výsledné soubory získané pomocí parsování dat:

- jmeno_organisumu_Promotor1000.fna
- jmeno_organisumu_Promotor500.fna
- jmeno_organisumu_Promotor300.fna
- jmeno_organisumu_Offgene1000.fna
- jmeno_organisumu_Offgene500.fna
- jmeno_organisumu_Offgene300.fna

– jmeno_organisumu_CDS.fna

Na konec tedy každá složka s organismem obsahovala kromě 2 stažených souborů dalších 7 s naparsovanými daty.

2.1.3 Tvorba datasetu

Data jsou rozdělena na trénovací a validační. Validační data jsou brána z organismu Saccharomyces cerevisiae, trénovací ze všech ostatních organismů.

2.1.3.1 Trénovací data

Pomocí pythonu bylo vytvořeno 7 souborů, které obsahovali všechny sekvence ze všech organismů kromě organismu Saccharomyces cerevisiae. ten byl použit následně pro validační data. Vzniklo tedy:

– AllOffgeneSeq300.fsa

– AllOffgeneSeq500.fsa

– AllOffgeneSeq1000.fsa

– AllPromotorSeq300.fsa

– AllPromotorSeq500.fsa

– AllPromotorSeq1000.fsa

– AllCDSeq.fsa

Pro následné trénovaní je vždy vzatá trojice těchto souboru a to vždy AllCDSeq.fsa a AllPromotorSeq s AllOffgeneSeq, které mají stejné číslo (stejná dálku promotoru od CDS).

Pomocí pythonu jsou z těchto souborů získány různé délky sekvencí a to o délce 300,100 a 50. Algoritmus pro extrakci funguje tím způsobem, že bere všechny sekvence daného souboru a naseká je na menší o dané délce. Vysekávaní probíhá posouváním okna o délce výstupní sekvence s krokem rovnající se též délky, tudíž se výsledné sekvence nepřekrývají. Sekvence se uloží poté do paměti v podobě 3 python seznamů (list). Z důvodu, že model rozumí jen číslům, musí být sekvence před vstupem do modelu převedena na čísla. Převod na čísla, který byl použit vypadá takto:

– báze A $\rightarrow$ 0

– báze T $\rightarrow$ 1

- báze C → 2
- báze G → 3

2.1.3.2 Validační data

Zde bylo vytvořeno též 7 souborů obsahující všechny sekvence z organismu Saccharomyces cerevisi

Příprava dat na validaci modelu funguje stejně jako pro trénovací data.

2.1.4 Úprava datasetu pomocí blast+

Z důvodu zlepšení datasetu, aby trénovací dataset neobsahoval stejná nebo podobná data jako validační, zde bude použit nástroj blast+.

Blast+ (Basic Local Alignment Search Tool Plus) je sada programů používaných k porovnávání biologických sekvencí, jako jsou DNA a proteiny. Tento software umožňuje hledat podobnosti mezi různými sekvencemi.

Nejdříve aby mohly být prohledány naše data se z nich musí udělat databáze. Na to byl použit blast+ příkaz blast makeblastdb. Z každého našeho souboru z daty byla tedy vytvořena databáze. Následně byl použit příkaz blastn, kterým jsme porovnávali jednotlivé validační soubory s databázemi. Výstup je textový soubor obsahující nálezy podobných sekvencí. Každý tento nález má u sebe velikost sekvence podobnosti (aligment length) a procento podobnosti.

Pomocí python skriptu jsme jednotlivé sekvence, které byly podobné a jejich parametr aligment length byl větší jak 100 vyhodily a do nového vyfiltrovaného souboru uložily ty zbylé. Vzniklo tedy nových 7 souborů každý menší o vyfiltrované sekvence něž jeho předchůdce. Například velikost starého souboru s promotory o délce 1000 měl 15,54 GB a ten nový 15,53 GB.

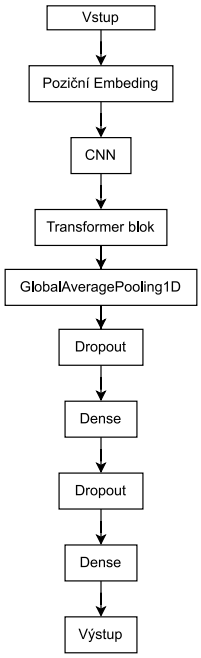
2.2 Model

Model použitý v této práci představuje komplexní architekturu, která kombinuje několik klíčových vrstev a mechanismů pro zpracování dat. Zde je menší popis jednotlivých částí modelu:

1. Vstup modelu je vektor s čísly o délce 300/100/50
2. Embedding vrstva: Na začátku modelu je embedding vrstva, která převede vstupní sekvence čísel, na kontinuální vektorové reprezentace. Tato reprezentace zachycuje sémantiku a vztahy mezi jednotlivými prvky vstupu.
3. Následuje konvoluční vrstva, která se specializuje na detekci lokálních vzorů a struktur v datech. Pomocí konvolucí dokáže model zachytit různé úrovně abstrakce a vzory ve vstupních datech.

4. Transformer blok, který umožňuje modelu zachytávat vztahy mezi prvky. Jeho implementace je převzata ze stránky keras [28].
5. Po transformaci vstupu konvolucí a Transformerem následuje global average pooling. Tato vrstva provádí průměrnou agregaci všech hodnot vstupu. To pomáhá modelu se zaměřit na celkový kontext dat a zároveň snižuje rozměr dat.
6. Proti přetrénování modelu následuje dropout vrstva.
7. Dále je hustá (dense) vrstva, která umožňuje modelu hledat komplexní vztahy mezi daty.
8. Poté další dropout vrstva.
9. Na závěr modelu je umístěna další hustá (dense) vrstva, která transformuje získané informace z předchozích vrstev na výstup.
10. Výstup modelu je poté vektor obsahující 3 pravděpodobnosti, že daná sekvence patří k jednotlivým třídám.

Implementace všech ostatní částí modelu, kromě transformeru, jsou převzaty z Python knihovny TensorFlow [29]



Obrázek 12: Architektura modelu

2.3 Výsledky

Sekce bude obsahovat výsledky modelů s různými parametry a jejich porovnání.

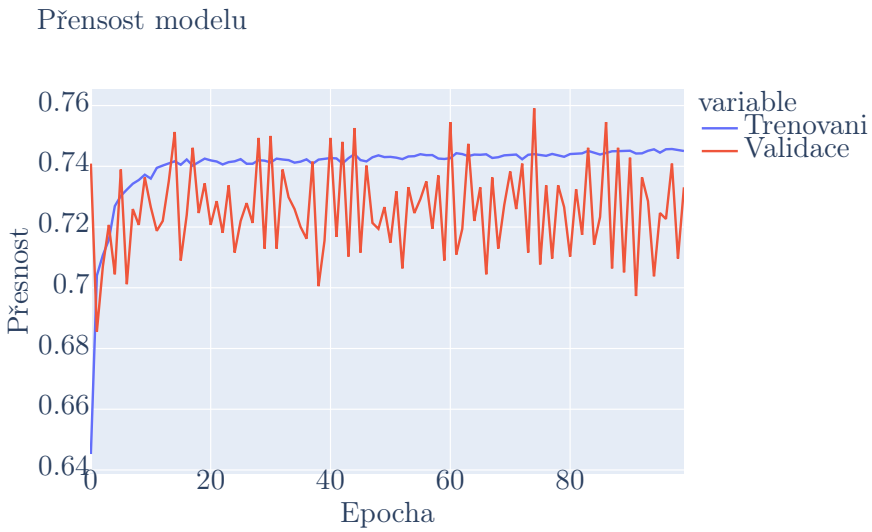
2.3.1 První model

První model, který zde bude ukázán byl trénován s parametry:

2.3.1.1 Parametry prního modelu

- Vstup obsahuje promotory, které nezasahují do jiných CDS : ANO
- Vyfiltrovaná data: ANO
- Velikost promotoru: 1000
- Velikost vstupní sekvence: 300
- Vyfiltrovaná data: ANO
- Dimenze embedingu: 200
- Velikost slovníku (vocab size): 4
- CNN : (výstupní dimenze: 200, velikost kernelu: 2, posun (stride): 1)
- CNN : (výstupní dimenze: 100, velikost kernelu: 7, posun (stride): 6)
- Dropout transformeru: 0.2
- Dimenze feed forward network v transformeru: 100
- Počet vrstev transformeru: 2
- Počet hlav transformeru: 2
- Dropout1: 0.6
- Dense layer: (dimenze výstupu: 80, aktivační funkce: relu [30])
- Dropout2: 0.6
- Výstupní dense layer: (dimenze výstupu: 3, aktivační funkce: softmax [31])
- Learning rate: 0.001
- Optimalizační algoritmus learning ratu: adam [32]
- délka trénování (epoch) : 100
- počet sekvencí na jednu epochu : 120000

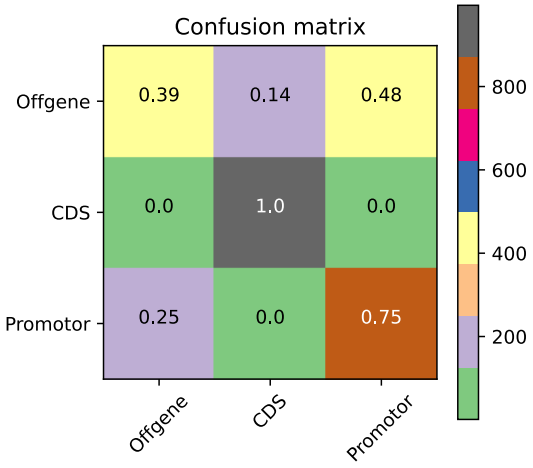
2.3.1.2 Výsledky prvního modelu



Obrázek 13: Výsledky trénování prvního modelu

Z grafu můžeme vidět, že přesnost modelu na trénovacích datech byla nějakých 74,5% a na validačních 73,31%

Confusion matice:



Obrázek 14: Confusion matice prvního modelu

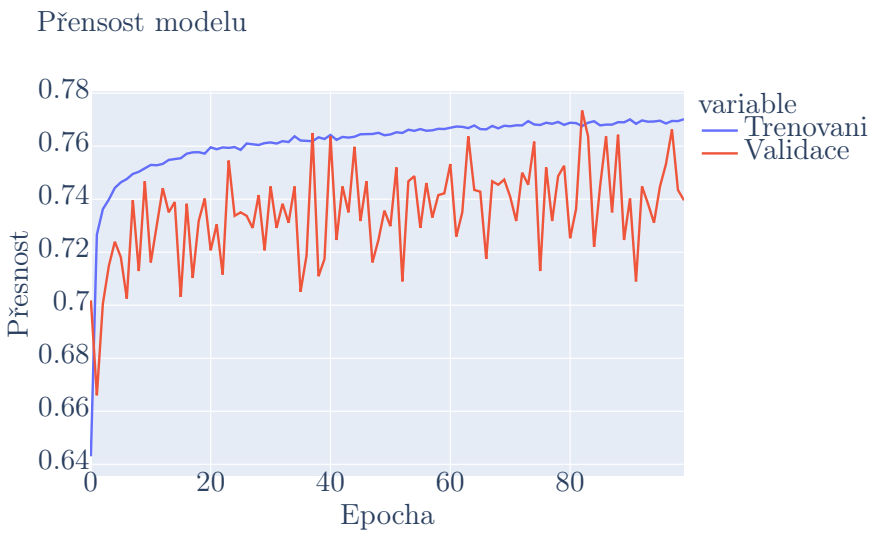
Z matice vidíme, že model velice dobře rozpoznává CDS, což je věc, která se dala čekat kvůli kodonům, které CDS obsahuje. Promotor už je rozpoznáván hůře ale stále lépe něž offgene.

2.3.2 Druhý model

2.3.2.1 Parametry druhého modelu

zde byl změněna od minulého modelu jen Dimenze feed forward networku u transfomrmeru ze 100 → 300

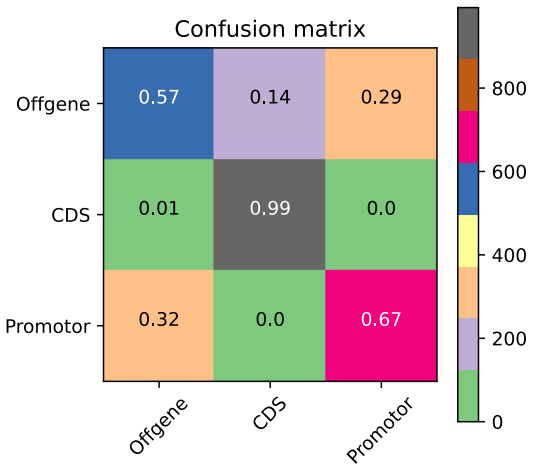
2.3.2.2 Výsledky druhého modelu



Obrázek 15: Výsledky trénování druhého modelu

Zde můžeme vidět zlepšení. Přesnost modelu na trénovacích datech byla nějakých 77% a na validačních 73,96%

Confusion matice:



Obrázek 16: Confusion matice druhého modelu

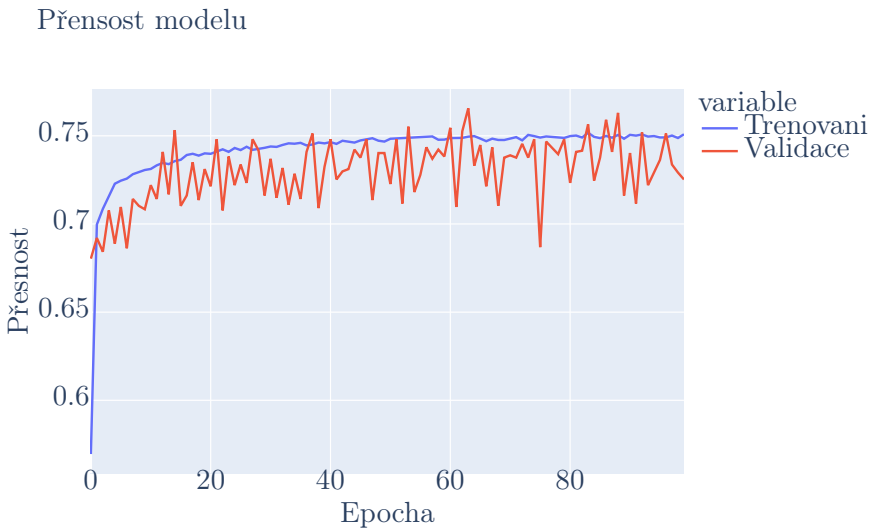
Z confusion matice je vidět, že model se zlepšil v predikci offgenu ale o něco zhoršil u promotoru.

2.3.3 Třetí model

2.3.3.1 Parametry třetího modelu

Zde byla změněna od minulého modelu jen dimenze výstupu dense layeru ze 80 → 20

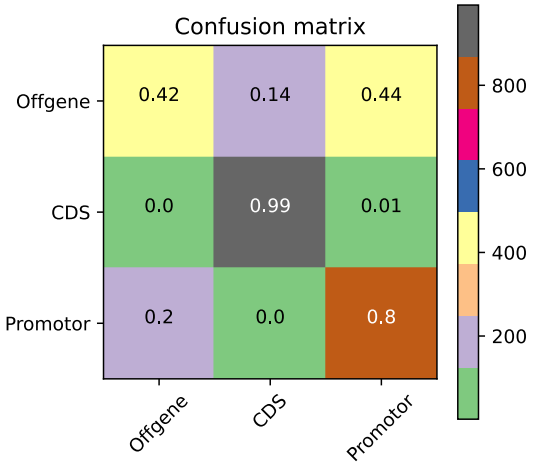
2.3.3.2 Výsledky třetího modelu



Obrázek 17: Výsledky trénování třetího modelu

U tohoto modelu došlo ke zhoršení. Přesnost modelu na trénovacích datech byla nějakých 75,09% a na validačních 72.53%

Confusion matice:



Obrázek 18: Confusion matice třetího modelu

Zde došlo k dalšímu zlepšení u promotoru a zhoršení u offgenu.

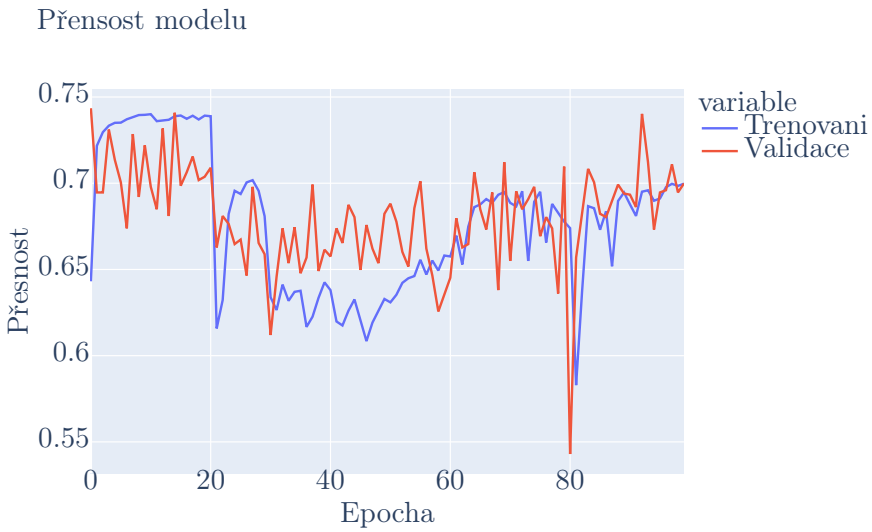
2.3.4 Čtvrtý model

2.3.4.1 Parametry čtvrtého modelu

zde byl změněna od minulého modelu:

- Počet vrstev transformeru: 2 → 4
- Dense layer: (dimenze výstupu vrácena: 20 → 80)

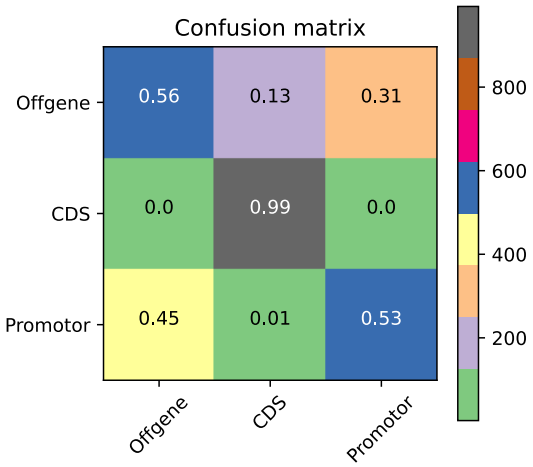
2.3.4.2 Výsledky čtvrtého modelu



Obrázek 19: Výsledky trénování čtvrtého modelu

Přesnost tohoto modelu na trénovacích datech byla nějakých 70% a na validačních také 70%

Confusion matice:



Obrázek 20: Confusion matice čtvrtého modelu

Když bychom porovnali confusin matici s confusin maticí druhého modelu, tak je zde vidět, že se výrazně zhoršila predikce Promotoru.

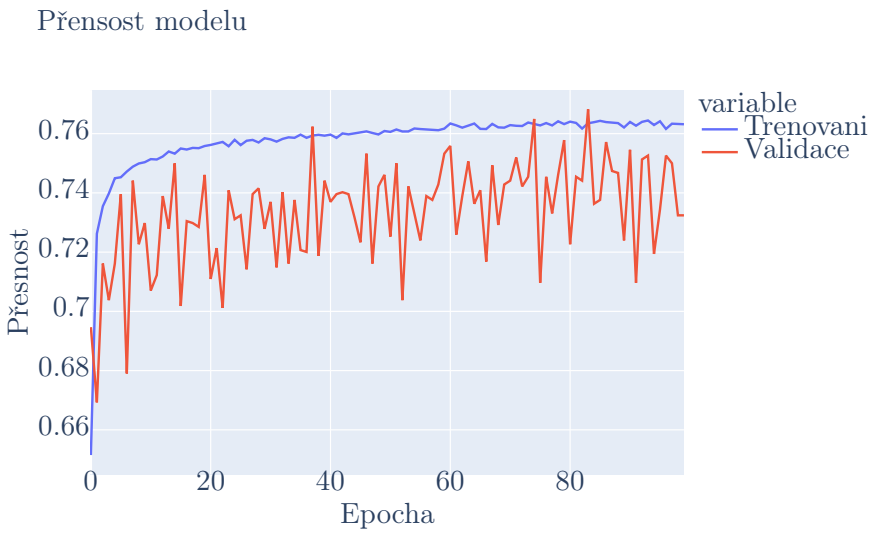
2.3.5 Pátý model

2.3.5.1 Parametry pátého modelu

zde byl změněna od minulého modelu:

- Počet vrstev transformeru vráceno: 4 → 2
- Počet hlav transformeru: 2 → 4

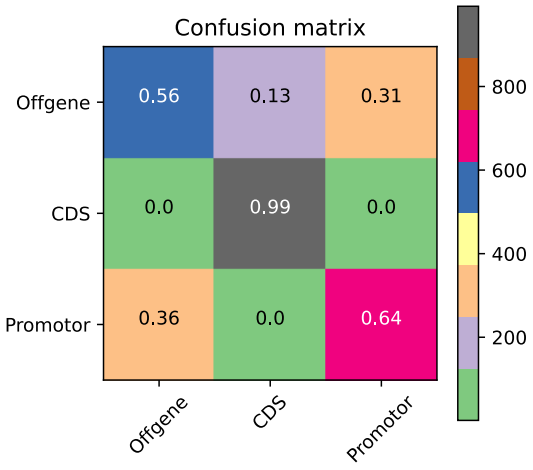
2.3.5.2 Výsledky pátého modelu



Obrázek 21: Výsledky trénování pátého modelu

Přesnost tohoto modelu na trénovacích datech byla nějakých 76,31% a na validačních 73,7%

Confusion matice:



Obrázek 22: Confusion matice pátého modelu

Zde oproti minulému modelu můžeme vidět zlepšení u promotoru.

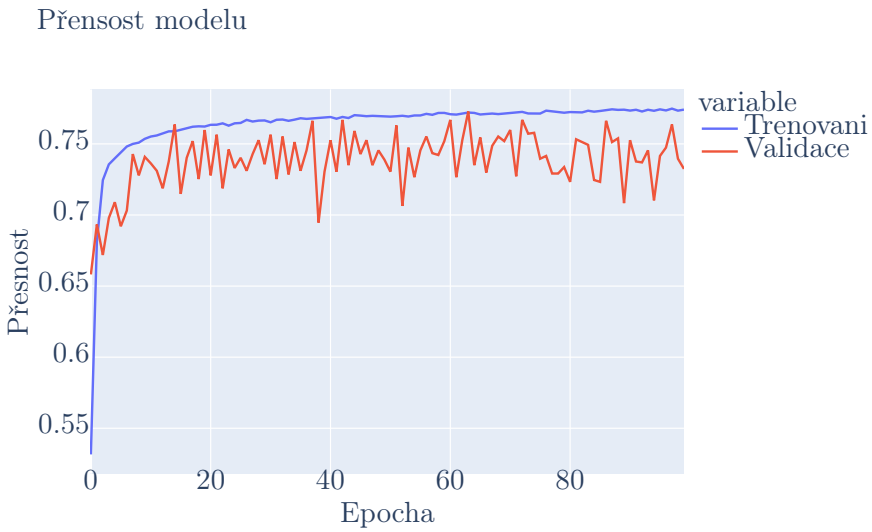
2.3.6 Šestý model

2.3.6.1 Parametry šestého modelu

zde bylo změněno od minulého modelu:

- Počet hlav transformeru vráceno: 4 → 2
- změna learning ratu : 0.001 → plánovač rychlosti učení (implementace převzata ze stránky Tensorflow [33])

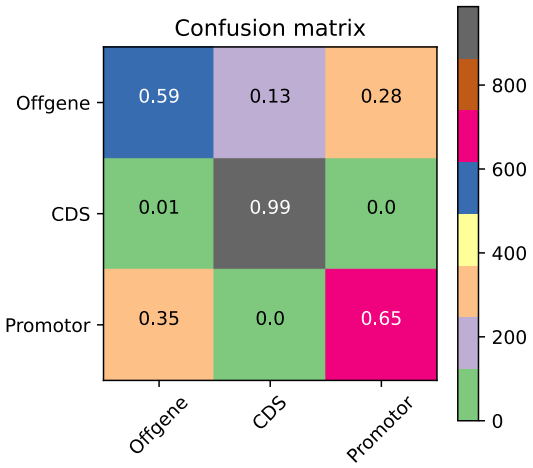
2.3.6.2 Výsledky šestého modelu



Obrázek 23: Výsledky trénování šestého modelu

Přesnost tohoto modelu na trénovacích datech byla nějakých 77,41% a na validačních také 73,24%

Confusion matice:



Obrázek 24: Confusion matice šestého modelu

2.3.7 Porovnání modelů s délkou vstupu 50 a odlišnou velikostí promotoru

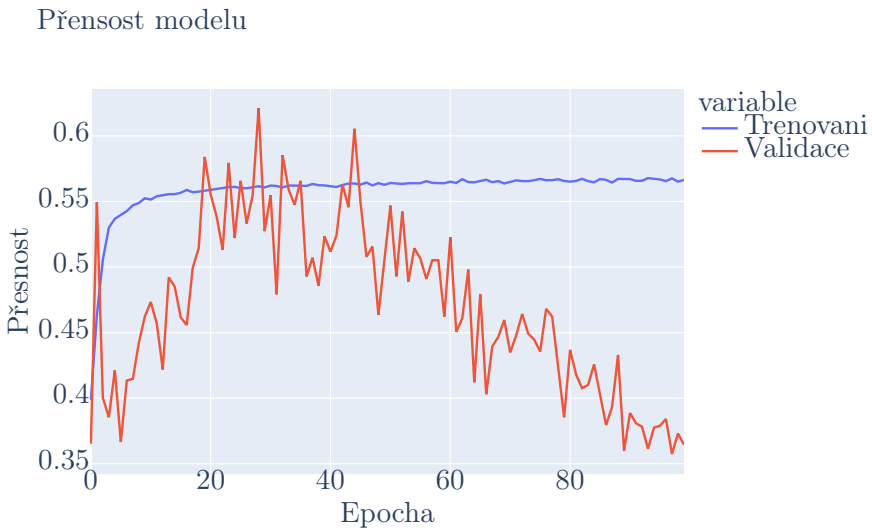
2.3.7.1 parametry modelů

Všechny tři modely měli tyto parametry stejné:

– Vstup obsahuje promotory, které nezasahují do jiných CDS : ANO

- Vyfiltrovaná data: ANO
- Velikost vstupní sekvence : 50
- Dimenze embedingu : 200
- Velikost slovníku (vocab size) : 4
- CNN : (výstupní dimenze: 200, velikost kernelu: 2, posun (stride): 1)
- CNN : (výstupní dimenze: 100, velikost kernelu: 7, posun (stride): 6)
- Dropout transformeru : 0.2
- Dimenze feed forward network v transformeru: 100
- Počet vrstev transformeru: 2
- Počet hlav transformeru: 2
- Dropout1 : 0.6
- Dense layer: (dimenze výstupu: 20, aktivační funkce: relu [30])
- Dropout2 : 0.6
- Výstupní dense layer: (dimenze výstupu: 3, aktivační funkce: softmax [31])
- Learning rate: 0.001
- Optimalizační algoritmus learning ratu: adam [32]
- délka trénování (epoch) : 100
- počet sekvencí na jednu epochu : 120000

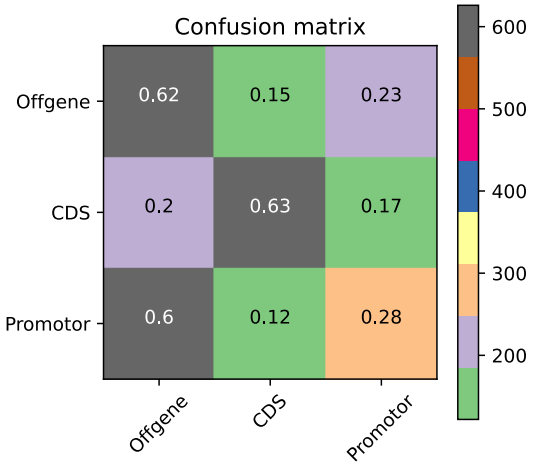
2.3.7.2 Výsledky modelu s velikostí promotoru 1000



Obrázek 25: Výsledky trénování modelu s velikostí promotoru 1000 a vstupu 50

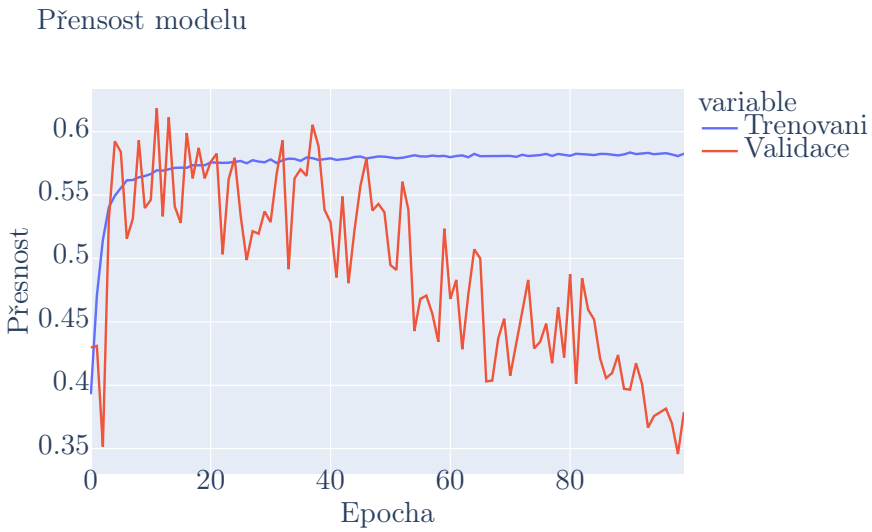
Zde je vidět, že snížení velikosti vstupu velice zasáhlo přesnost modelu. Přesnost tohoto modelu na trénovacích datech byla jen nějakých 56,64% a na validačních 36,46%

Confusion matice:



Obrázek 26: Confusion matice modelu s velikostí promotoru 1000 a vstupu 50

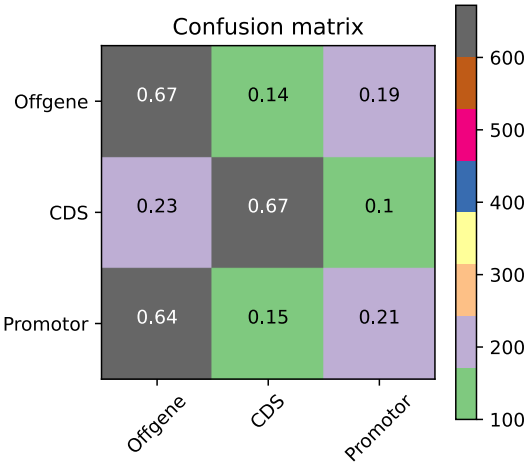
2.3.7.3 Výsledky modelu s velikostí promotoru 500



Obrázek 27: Výsledky trénování modelu s velikostí promotoru 500 a vstupu 50

Snížení velikosti promotoru zde trochu pomohlo. Přesnost tohoto modelu na trénovacích datech byla nějakých 58,26% a na validačních 37,89%

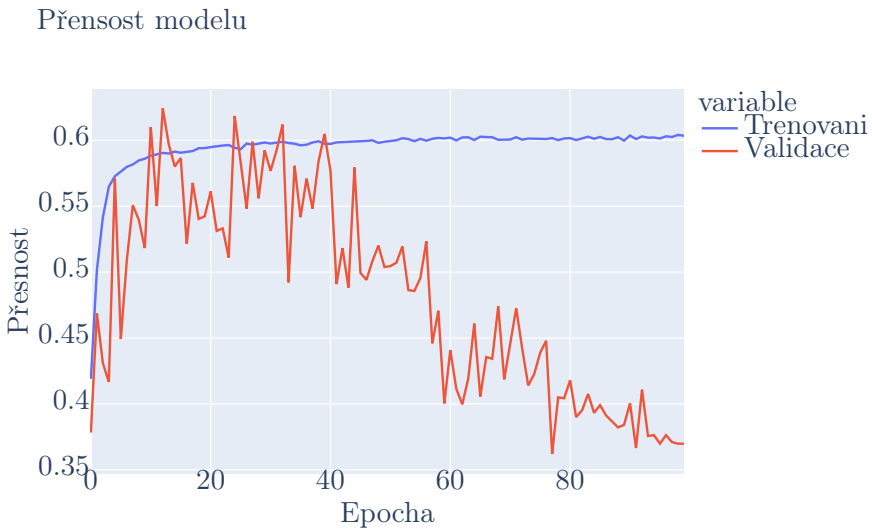
Confusion matice:



Obrázek 28: Confusion matice modelu s velikostí promotoru 500 a vstupu 50

Zde oproti minulému modelu můžeme vidět minimální zlepšení u offgenu a CDS. Ale z pohledu všech modelů tento model patří k těm nejhorším.

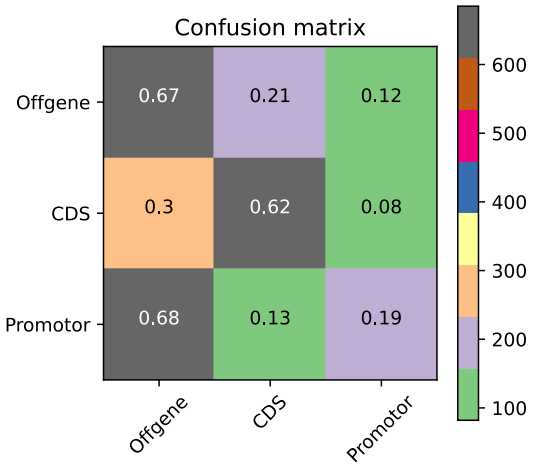
2.3.7.4 Výsledky modelu s velikostí promotoru 300



Obrázek 29: Výsledky trénování modelu s velikostí promotoru 300 a vstupu 50

Snížení velikosti promotoru na 300 zde trochu pomohlo jen u přesnosti na trénovacích datech. Přesnost tohoto modelu na trénovacích datech byla nějakých 60,35% ale na validačních byla menší a to jen 39,98%

Confusion matice:



Obrázek 30: Confusion matice modelu s velikostí promotoru 300 a vstupu 50

2.3.7.5 Shrnutí

Tabulka 1: Tabulka ohodnocení modelů se vstupem o délce 50 a různými velikostmi promotoru

	Promotor 1000	Promotor 500	Promotor 300
Validační data	36,46%	37,89%	39,98%
Trénovací data	56,64%	58,26%	60,35¨%

Jak můžeme vidět z tabulky, nejlépe zde vzhledem k validačním datům dopadl model s velikostí promotoru 300. Ale i tak vstup o délce 50 dosahuje velice malé přesnosti u všech velikostí promotorů.

2.3.8 Porovnání modelů s délkou vstupu 300 a odlišnou velikostí promotoru

2.3.8.1 Parametry modelů

Všechny 3 modely obsahují tyto parametry:

- Vstup obsahuje promotory, které nezasahují do jiných CDS : ANO
- Vyfiltrovaná data: ANO
- Velikost vstupní sekvence : 50
- Dimenze embedingu : 200
- Velikost slovníku (vocab size) : 4
- CNN : (výstupní dimenze: 200, velikost kernelu: 2, posun (stride): 1)
- CNN : (výstupní dimenze: 100, velikost kernelu: 7, posun (stride): 6)
- Dropout transformeru : 0.2
- Dimenze feed forward network v transformeru: 100
- Počet vrstev transformeru: 2
- Počet hlav transformeru: 2
- Dropout1 : 0.6
- Dense layer: (dimenze výstupu: 20, aktivační funkce: relu [30])
- Dropout2 : 0.6
- Výstupní dense layer: (dimenze výstupu: 3, aktivační funkce: softmax [31])

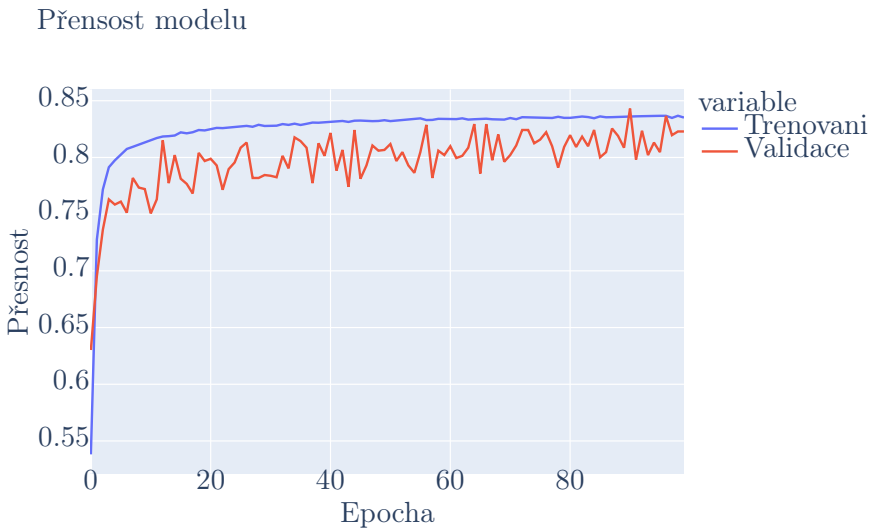
– Learning rate: 0.001

– Optimalizační algoritmus learning ratu: adam [32]

– délka trénování (epoch) : 100

– počet sekvencí na jednu epochu : 120000

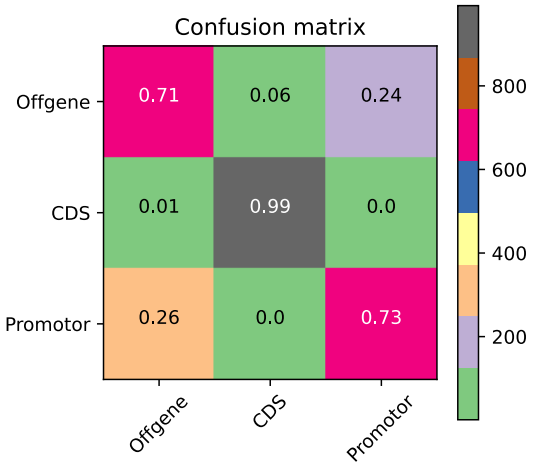
2.3.8.2 Výsledky modelu s délkou promotoru 300



Obrázek 31: Výsledky trénování modelu s délkou promotoru 300 a vstupem 300

Z grafu můžeme vidět, že tento model dosahuje jedny z nejlepších hodnot. Přesnost tohoto modelu na trénovacích datech byla nějakých 83,52% a na validačních 82,29%.

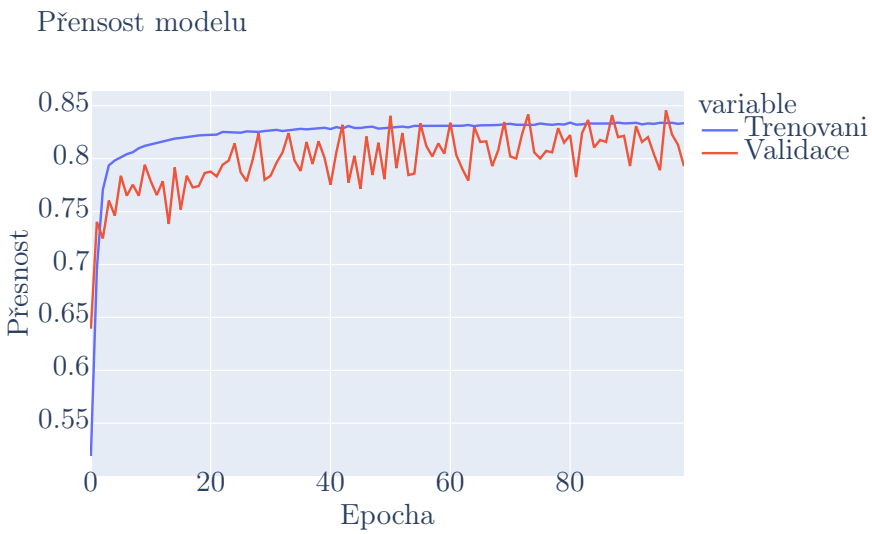
Confusion matice:



Obrázek 32: Confusion matice modelu s délkou promotoru 300 a vstupem 300

Confusion matice jen potvrzuje tvrzení výše. Tento model je zatím nejlepší.

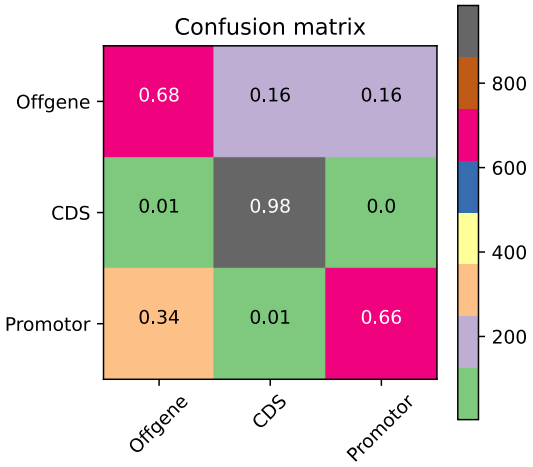
2.3.8.3 Výsledky modelu s délkou promotoru 500



Obrázek 33: Výsledky trénování modelu s délkou promotoru 500 a vstupem 300

Přesnost tohoto modelu na trénovacích datech byla nějakých 83,35% a na validačních 79,3%.

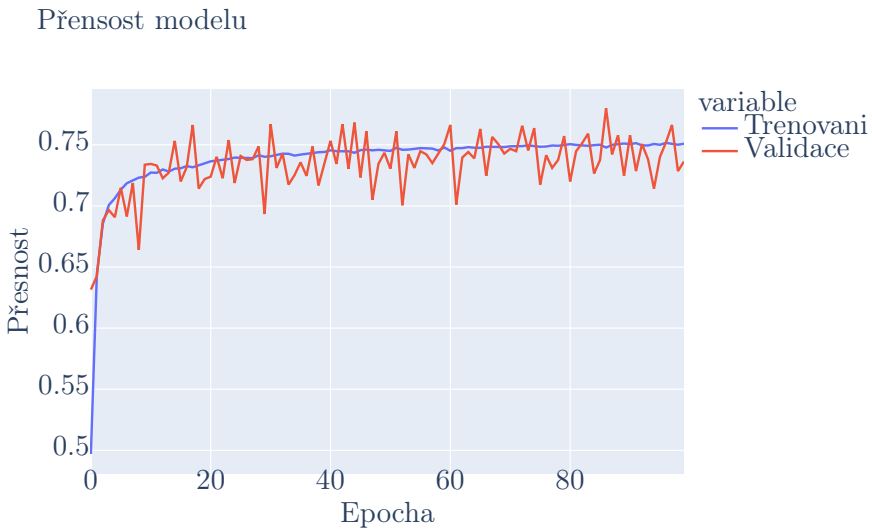
Confusion matice:



Obrázek 34: Confusion matice modelu s délkou promotoru 500 a vstupem 300

Zde vidíme, že se model o něco zhoršil a to hlavně u rozpoznávání promotoru a offgenu.

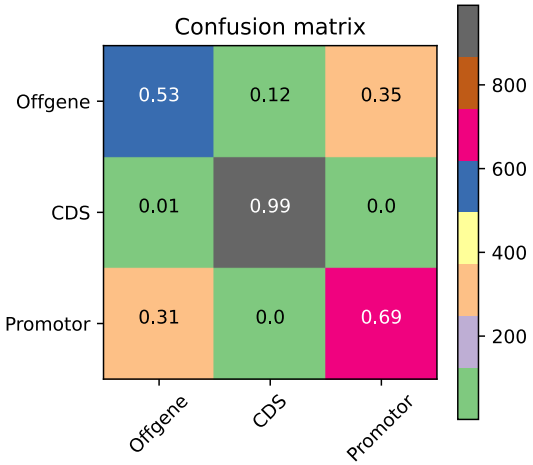
2.3.8.4 Výsledky modelu s délkou promotoru 1000



Obrázek 35: Výsledky trénování modelu s délkou promotoru 1000 a vstupem 300

Přesnost tohoto modelu na trénovacích datech byla nějakých 75,09% a na validačních byla 73,63%.

Confusion matice:



Obrázek 36: Confusion matice modelu s délkou promotoru 500

Confusion matice zde ukazuje zhoršení a to u rozpoznaní offgenu.

2.3.8.5 Shrnutí

Tabulka 2: Tabulka ohodnocení modelů se vstupem o délce 300 a různými velikostmi promotoru

	Promotor 1000	Promotor 500	Promotor 300
Validační data	76,63%	79,3%	82,29%
Trénovací data	75,09%	83,35%	83,52%

Z tabulky zde vidíme, že model s velikostí promotoru 300 dosahuje nejlepších hodnot v celé práci. nejhorší je zde model s délkou promotoru 1000.

2.3.9 Porovnání modelů s délkou vstupu 100 a odlišnou velikostí promotoru

2.3.9.1 Parametry modelů

Všechny 3 modely obsahují tyto parametry:

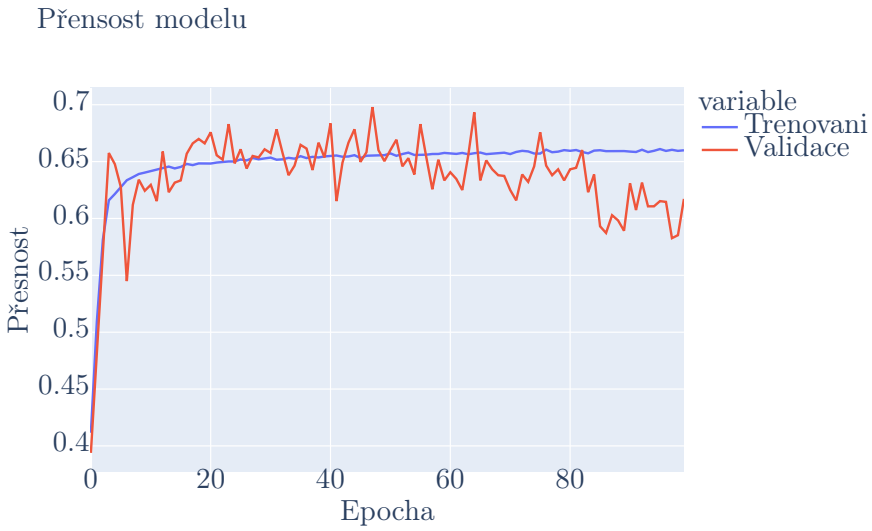
- Vstup obsahuje promotory, které nezasahují do jiných CDS : ANO
- Vyfiltrovaná data: ANO
- Velikost vstupní sekvence : 100
- Dimenze embedingu : 200
- Velikost slovníku (vocab size) : 4
- CNN : (výstupní dimenze: 200, velikost kernelu: 2, posun (stride): 1)
- CNN : (výstupní dimenze: 100, velikost kernelu: 7, posun (stride): 6)
- Dropout transformeru : 0.2
- Dimenze feed forward network v transformeru: 100
- Počet vrstev transformeru: 2
- Počet hlav transformeru: 2
- Dropout1 : 0.6
- Dense layer: (dimenze výstupu: 20, aktivační funkce: relu [30])
- Dropout2 : 0.6
- Výstupní dense layer: (dimenze výstupu: 3, aktivační funkce: softmax [31])
- Learning rate: 0.001

– Optimalizační algoritmus learning ratu: adam [32]

– délka trénování (epoch) : 100

– počet sekvencí na jednu epochu : 120000

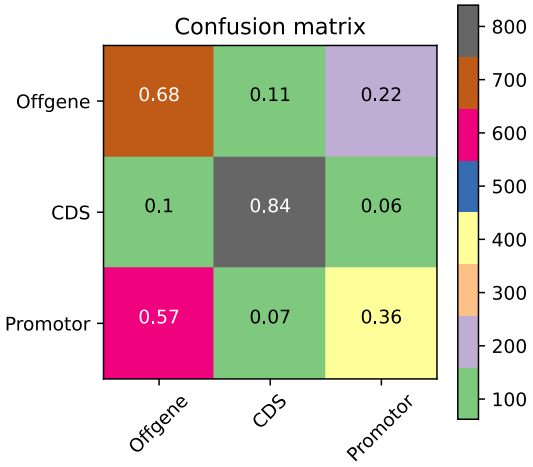
2.3.9.2 Výsledky modelu s délkou promotoru 300



Obrázek 37: Výsledky trénování modelu s délkou promotoru 300 a vstupem 100

Přesnost tohoto modelu na trénovacích datech byla nějakých 66% a na validačních byla 61,72%.

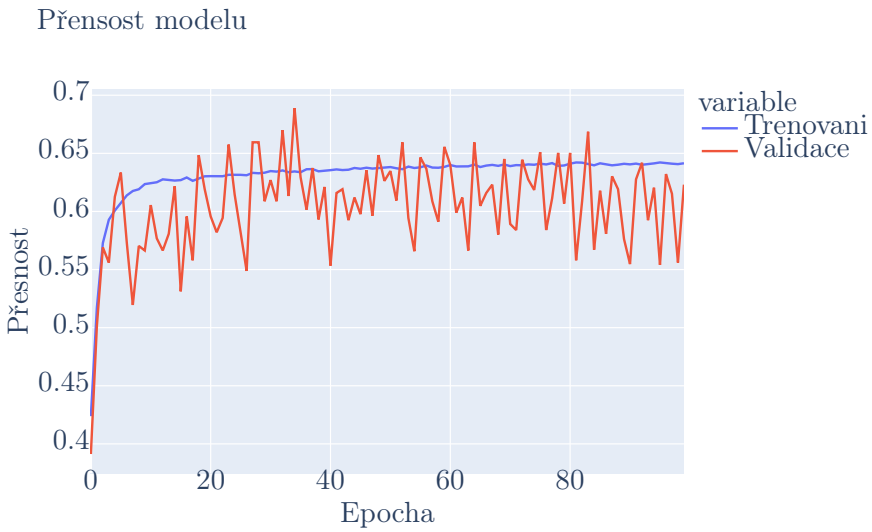
Confusion matice:



Obrázek 38: Confusion matice modelu s délkou promotoru 300 a vstupem 100

Z confusion matice můžeme vidět, že vstup o délce 100 poznamenal přesnost rozpoznání u promotoru a CDS.

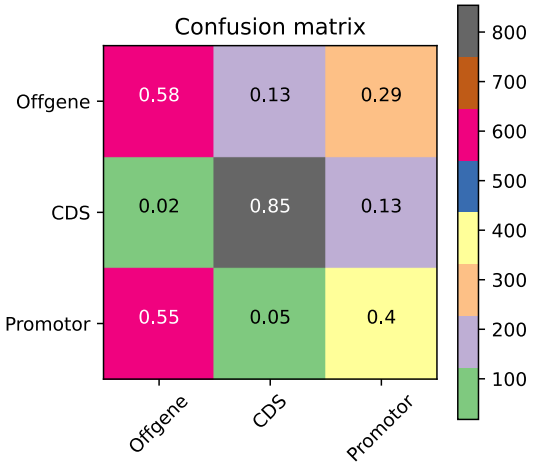
2.3.9.3 Výsledky modelu s délkou promotoru 500



Obrázek 39: Výsledky trénování modelu s délkou promotoru 500 a vstupem 100

Přesnost tohoto modelu na trénovacích datech byla nějakých 64.15% a na validačních byla 62,23%.

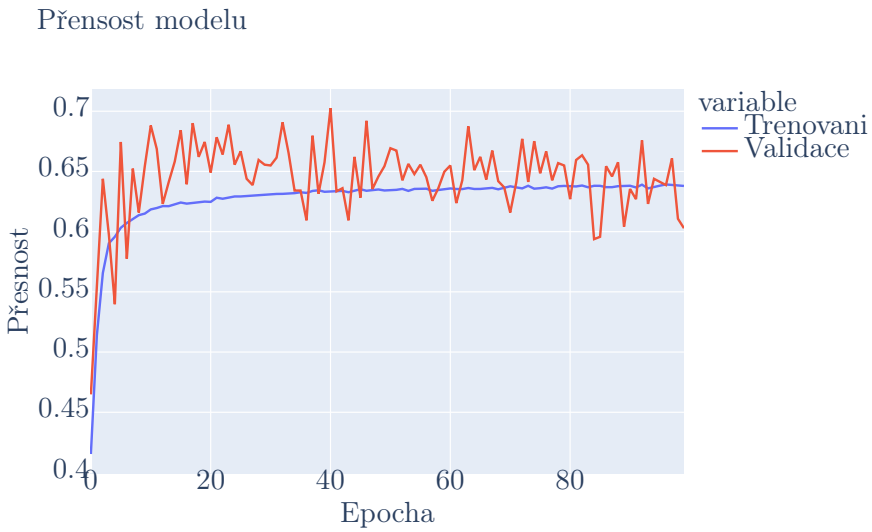
Confusion matice:



Obrázek 40: Confusion matice modelu s délkou promotoru 500 a vstupem 100

Confusion matice ukazuje menší zlepšení modelu u rozpoznaní promotoru a CDS ale zhoršení u offgenu.

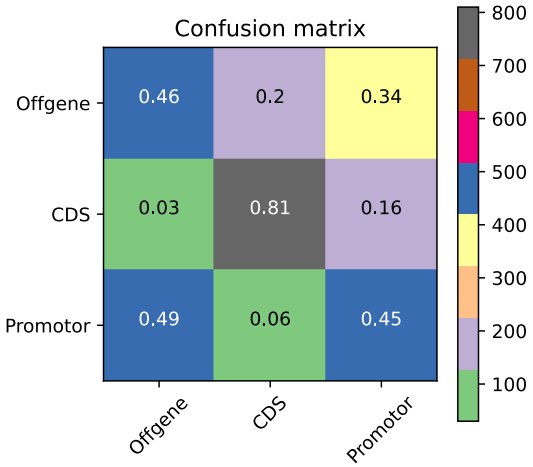
2.3.9.4 Výsledky modelu s délkou promotoru 1000



Obrázek 41: Výsledky trénování modelu s délkou promotoru 1000 a vstupem 100

Přesnost tohoto modelu na trénovacích datech byla nějakých 63.79% a na validačních byla 60,29%.

Confusion matice:



Obrázek 42: Confusion matice modelu s délkou promotoru 1000 a vstupem 100

Z dat confusion matice zde opět došlo ke zlepšení rozpoznávání promotoru ale zhoršení jak u Offgenu tak i u CDS.

2.3.9.5 Shrnutí

Tabulka 3: Tabulka ohodnocení modelů se vstupem o délce 100 a různými velikostmi promotoru

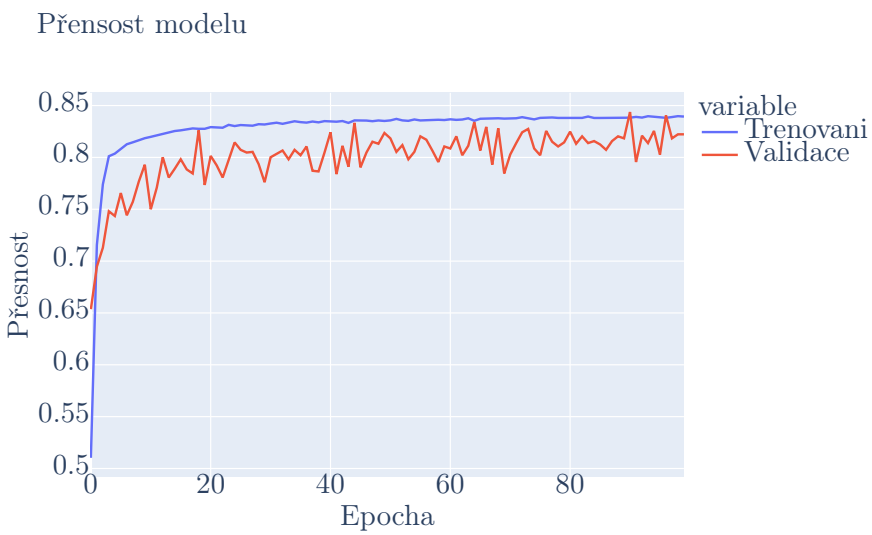
	Promotor 1000	Promotor 500	Promotor 300
Validační data	60,29%	62,23%	61,72%
Trénovací data	63,79%	64,15%	66¨%

Z tabulky je zde vidět, že nejlepší přesnost na validačních datech měl tentokrát model s velikostí promotoru 500. Při celkovém porovnání s ostatními vstupy se tato délka vstupu 100 umístila přesně mezi vstupy 300 a 50. Nejlepší vstup je tedy o délce 300

2.3.10 Porovnání modelů s vyfiltrovanými a nevyfiltrovanými trénovacími daty

Pro porovnání zde bude natrénovat model se stejnými parametry jako model, který dosahoval nejlepších hodnot 2.3.8.2

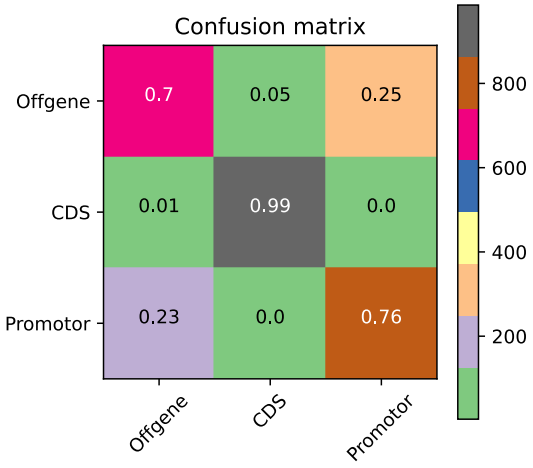
2.3.10.1 Výsledky modelu s nevyfiltrovanými trénovacími daty



Obrázek 43: Výsledky trénování modelu s nevyfiltrovanými trénovacími daty

V porovnání s modelem s vyfiltrovanými trénovacími daty tento model dosahuje lepší přesnosti na trénovacích datech a to z 83,52% → 83,94%. Ale u validačních dat je to naopak. Zde tento model zaostává ale ne o 0,05%.Přesnost tohoto modelu na validačních datech se zhoršila z 82,29% → 82,23%

Confusion matice:



Obrázek 44: Confusion matice modelu s nevyfiltrovanými trénovacími daty

Pokud bychom porovnali obě confusion matice, tak lepší výsledek dosahuje tento model s nevyfiltrovanými trénovacími daty. To může mít například příčinu v tom, že pro tvorbu confusion matice se bere vždy jen část validačních dat.

2.3.10.2 Shrnutí

Celkově jde tedy říci, že filtrace trénovacích dat má nějaký účinek na výsledný model, ale tento účinek v tomto případě není nijak velký

2.4 Program na rozpoznání části genomu

Sekce obsahuje popis skriptu sloužícího pro real-time vizualizaci rozpoznávání částí DNA.

2.4.1 Funkce programu

Program slouží pro real-time zobrazování predikcí na graf ve webovém prohlížeči. Vstup tohoto programu je fna soubor se sekvencemi nukleových bází, identifikátor chromozomu, ze které má být vybrána sekvence a index začátku a konce sekvence (range), kterou chceme rozpoznat. Rozpoznáváno je vždy okno o délce 300, které se posouvá o 1 bázi dál až dojde na konec sekvence. Daná výstupní pravděpodobnost je poté přidělena 150. bázi v sekvenci. Aby byly rozpoznány všechny báze v dané range, je potřeba vzít vybranou sekvenci která bude začínat o 149 bází dříve něž je určeno uživatelem a končit o 150 bází dál. Výstup je graf, který na ose x obsahuje jednotlivé indexy, které byly zadány jako vstup a osa y vyznačuje pravděpodobnost, že daný index (sekvence o délce 300, která se nachází okolo tohoto indexu) se klasifikuje do jedné ze 3 tříd (CDS, Promotor, Offgene)

2.4.2 Implementace

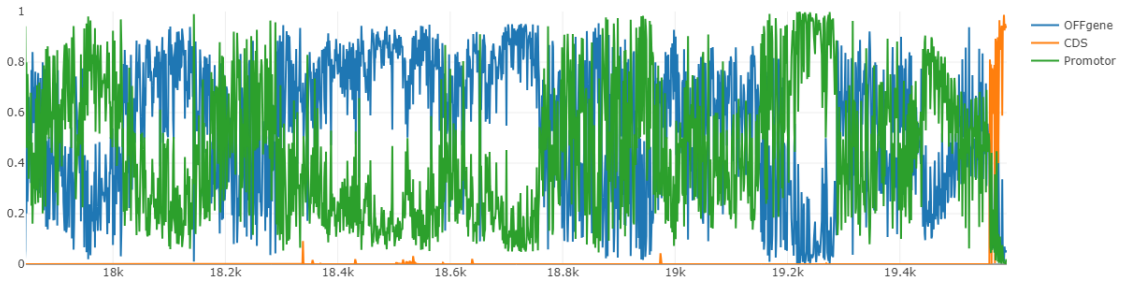
Skript byl vytvořen pomocí pythonu a jeho knihoven:

- Plotly - slouží k vykreslování grafu.
- Dash - slouží pro tvorbu interaktivních webových aplikací. Zde použito pro aktualizace grafu.
- Tensorflow - slouží pro načtení natrénovaného modelu.
- Fastaparser - slouží pro jednodušší práci s fna soubory.
- Numpy - slouží pro převedení sekvence na číselný vektor, kterému rozumí model.

2.4.3 Ukázka

Pomocí nástroje YeasTSS byla vybrána sekvence z organismu Saccharomyces cerevisiae o délce 1744 bází, která začíná ihned po konci jednoho CDS a končí na začátku dalšího CDS. Tato oblast se nachází v chromozomu 5 a index začátku a konec je 17845 a 19589 Očekávání zde je, že čím dál blíže se dostaneme k dalšímu CDS tím větší šance je, že se zde nachází promotor. Pro ukázku zde byl zvolen nejlepší model, ke kterému jsme v této práci došli 2.3.8.2 .

Výsledný graf:



Obrázek 45: Výstup z programu pro vizualizaci predikce modelu

Z výsledku vidíme, že model ze začátku chybně kvalifikoval do promotoru, což může být tím, že model nebyl natrénován na sekvence, které prolínají dvě kvalifikační třídy (zde offgene s CDS). Dále už kvalifikace probíhá skoro podle očekávání. Největší pravděpodobnost kvalifikace do promotoru model udává u indexů 19158 - 19281.

3 Závěr

V této práci jsme se věnovali úkolu rozpoznávání jednotlivých částí genomu, což představuje důležitý krok směrem k odhalení genetických tajemství. Kombinací metod analýzy dat a strojového učení jsme dosáhli zajímavých výsledků v identifikaci kódovacích sekvencí (CDS), promotorových oblastí a offgenu.

Provedli jsme porovnání modelů s různými parametry. Vyšlo nám, že nejlepší model je model, který byl trénován z dat s velikostí promotoru 300, což dává smysl protože čím blíže se nacházíme k CDS tím jasnější struktura promotoru je (nachází se zde sekvence s větší hojností nukeových bází T a A). Přesnost našeho nejlepšího modelu pro rozpoznávání částí genomu byla vyhodnocena na validačních datech s hodnotami 99% pro kódovací sekvence, 73% pro promotorové oblasti a 71% pro Offgene.

Nicméně, je důležité zdůraznit, že i přes tyto zajímavé hodnoty stále existuje prostor pro další zdokonalení. Komplexnost dat genomu a variabilita genetických struktur mohou mít vliv na přesnost rozpoznávání. Budoucí práce by měly směřovat k vývoji ještě sofistikovanějších modelů, které budou schopny lépe zohlednit tuto komplexitu a dosáhnout ještě vyšší přesnosti.

Dále byl vytvořen program na real-time vizualizaci predikce těchto modelů. V budoucnu by tento program mohl být ještě zlepšen aby zde nebylo nutné zadávat parametry do python souboru ale rovnou vše ovládat přes prohlížeč. Kvůli nápadu na toto zlepšení byl program rovnou psán pomocí python knihovny dash, která by měla tyto vylepšení v budoucnu dovolit.

Všechen kód použit na tuto práci je přístupný na githubu [34].

Seznam použité literatury

1. BUSTIN, Stephen. Molecular Biology of the Cell, Sixth Edition. *International Journal of Molecular Sciences*. 2015, roč. 16, č. 12, s. 28123–28125. Dostupné z DOI: 10.3390/ijms161226074.
2. BATES, Sarah A. *Deoxyribonucleic Acid (DNA)*. NaN. Dostupné také z: https://www.genome.gov/genetics-glossary/Deoxyribonucleic-Acid.
3. MEFANET; SR, síť lékařských fakult ČR a. *Sekundární struktura DNA – WikiSkripta* [online]. [cit. 2023-07-30]. Dostupné z: https://www.wikiskripta.eu/w/Sekund%C3%A1rn%C3%AD_struktura_DNA.
4. WHITFORD, David. *Proteins: Structure and Function*. 2013. Dostupné také z: https://books.google.cz/books?hl=cs&lr=&id=AnodNhuMAdkC&oi=fnd&pg=PT12&dq=Proteins:+Structure+and+Function&ots=WaeSxamH0-&sig=Ud_C1tsyRGIeW-oKHp1Ui4o6JVY&redir_esc=y#v=onepage&q=Proteins%3A%20Structure%20and%20Function&f=false.
5. ALLIANCE, Genetic; SCREENING SERVICES, The New York-Mid-Atlantic Consortium for Genetic; NEWBORN. Understanding Genetics. 2009. Dostupné také z: https://www.ncbi.nlm.nih.gov/books/NBK115563/.
6. *Promoter*. 2023. Dostupné také z: https://www.genome.gov/genetics-glossary/Promoter.
7. LE, Nguyen Quoc Khanh; YAPP, Edward Kien Yee; NAGASUNDARAM, N.; YEH, Hui-Yuan. Classifying Promoters by Interpreting the Hidden Information of DNA Sequences via Deep Learning and Combination of Continuous FastText N-Grams. *Frontiers in Bioengineering and Biotechnology*. 2019, roč. 7. Dostupné z DOI: 10.3389/fbioe.2019.00305.
8. WANG, Sun-Chong. Artificial Neural Network. In: *Interdisciplinary Computing in Java Programming*. Springer US, 2003, s. 81–100. Dostupné z DOI: 10.1007/978-1-4615-0377-4_5.
9. RUMELHART, David E.; HINTON, Geoffrey E.; WILLIAMS, Ronald J. Learning representations by back-propagating errors. *Nature*. 1986, roč. 323, č. 6088, s. 533–536. Dostupné z DOI: 10.1038/323533a0.
10. VASWANI, Ashish; SHAZEER, Noam; PARMAR, Niki; USZKOREIT; GOMEZ, Aidan N.; KAISER, Lukasz; POLOSUKHIN, Illia. Attention is All You Need. 2017.
11. ALAMMAR, Jay. *The Illustrated Transformer – Jay Alammar – Visualizing machine learning one concept at a time*. NaN. Dostupné také z: https://jalammar.github.io/illustrated-transformer/.
12. , sawan saxena sawan. *Understanding Embedding Layer in Keras | by sawan saxena | Analytics Vidhya | Medium*. 2020. Dostupné také z: https://medium.com/analytics-vidhya/understanding-embedding-layer-in-keras-bbe3ff1327ce.

13. SRINIVASAMURTHY, Ravisutha Sakrepatna. Understanding 1D Convolutional Neural Networks Using Multiclass Time-Varying Signals. 2018. Dostupné také z: https://tigerprints.clemson.edu/all_theses/2911.

14. H.L., Sneha. *2D Convolution in Image Processing - Technical Articles*. 2018. Dostupné také z: https://www.allaboutcircuits.com/technical-articles/two-dimensional-convolution-in-image-processing/.

15. S, Mohneesh. *Savitzky-Golay Filter for data Smoothing | by Mohneesh S | Towards AI*. 2022. Dostupné také z: https://pub.towardsai.net/savitzky-golay-filter-for-data-smoothing-3b7c1c5e7f69.

16. TEAM, Keras. *MaxPooling1D layer*. Dostupné také z: https://keras.io/api/layers/pooling_layers/max_pooling1d/.

17. JOSEPHINE, V L Helen; NIRMALA, A.P.; ALLURI, Vijaya Lakshmi. Impact of Hidden Dense Layers in Convolutional Neural Network to enhance Performance of Classification Model. *IOP Conference Series: Materials Science and Engineering*. 2021, roč. 1131, č. 1, s. 012007. Dostupné z DOI: 10.1088/1757-899x/1131/1/012007.

18. SRIVASTAVA, Nitish; HINTON, Geoffrey; KRIZHEVSKY, Alex; SUTSKEVER, Ilya; SALAKHUTDINOV, Ruslan. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*. 2014, roč. 15, č. 1, s. 1929–1958.

19. YING, Xue. An Overview of Overfitting and its Solutions. *Journal of Physics: Conference Series*. 2019, roč. 1168, s. 022022. Dostupné z DOI: 10.1088/1742-6596/1168/2/022022.

20. GRAVES, Alex; SCHMIDHUBER, Jürgen. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*. 2005, roč. 18, č. 5-6, s. 602–610. Dostupné z DOI: 10.1016/j.neunet.2005.06.042.

21. *Long Short-Term Memory Networks (LSTM)- simply explained! | Data Basecamp*. 2022. Dostupné také z: https://databasecamp.de/en/ml/lstms.

22. BREIMAN, Leo. Random Forests. *Machine Learning*. 2001, roč. 45, č. 1, s. 5–32. Dostupné z DOI: 10.1023/a:1010933404324.

23. *Rozhodovací stromy a chytré otázky – Základy informatiky pro střední školy*. Dostupné také z: https://popelka.ms.mff.cuni.cz/~lessner/mw/index.php/U%C4%8Debnice/Informace/Rozhodovac%C3%AD_stromy_a_chytr%C3%A9_ot%C3%A1zky.

24. N, Bhandari. *Comparison of machine learning and deep learning techniques in promoter prediction across diverse species*. PeerJ Computer Science, 2021. Dostupné z DOI: 10.7287/peerj-cs.365v0.2/reviews/3.

25. HAEUSSLER, Maximilian; ZWEIG, Ann S; TYNER, Cath; SPEIR, Matthew L; ROSENBLOOM, Kate R; RANEY, Brian J; LEE, Christopher M; LEE, Brian T; HINRICHS, Angie S; GONZALEZ, Jairo Navarro; GIBSON, David; DIEKHANS, Mark; CLAWSON, Hiram; CASPER, Jonathan; BARBER, Galt P; HAUSSLER, David; KUHN, Robert M; KENT, W James. The UCSC Genome Browser database: 2019 update. *Nucleic Acids Research*. 2018, roč. 47, č. D1, s. D853–D858. Dostupné z DOI: 10.1093/nar/gky1095.

26. HERNÁNDEZ, Daryl; JARA, Nicolás; ARAYA, Mauricio; DURÁN, Roberto E.; BUIL-ARANDA, Carlos. PromoterLCNN: A Light CNN-Based Promoter Prediction and Classification Model. *Genes*. 2022, roč. 13, č. 7. ISSN 2073-4425. Dostupné z DOI: 10.3390/genes13071126.

27. *Introduction to the Keras Tuner | TensorFlow Core*. Dostupné také z: https://www.tensorflow.org/tutorials/keras/keras_tuner.

28. TEAM, Keras. *Text classification with Transformer*. 2020. Dostupné také z: https://keras.io/examples/nlp/text_classification_with_transformer/.

29. *TensorFlow*.

30. *tf.keras.activations.relu | TensorFlow v2.13.0*. Dostupné také z: https://www.tensorflow.org/api_docs/python/tf/keras/activations/relu.

31. *tf.nn.softmax | TensorFlow v2.13.0* [online]. NaN. [cit. 2023-08-13]. Dostupné z: https://www.tensorflow.org/api_docs/python/tf/nn/softmax.

32. *tf.keras.optimizers.Adam | TensorFlow v2.13.0*. Dostupné také z: https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam.

33. *Neural machine translation with a Transformer and Keras | Text | TensorFlow*. Dostupné také z: https://www.tensorflow.org/text/tutorials/transformer#set_up_the_optimizer.

34. *kapaCZ/Rozdeleni-DNA-do-funkcnich-oblasti* [online]. [cit. 2023-08-14]. Dostupné z: https://github.com/kapaCZ/Rozdeleni-DNA-do-funkcnich-oblasti.