Deep Neural Network based Text-to-Speech
Beszédszintézis Mély Neurális Hálóval

Tamas Szanto
tmas.szanto@gmail.com

Gergely D. Nemeth
NeGeD.NG@gmail.com

Abstract

Human-like communication with the computers is a major application of Computer Science. The Text-to-Speech methods are significant part of the project. This paper is a review of the authors experience about the usefulness of the Deep Neural Networks' new fields regarding to the **TTS** problem.

Abstract

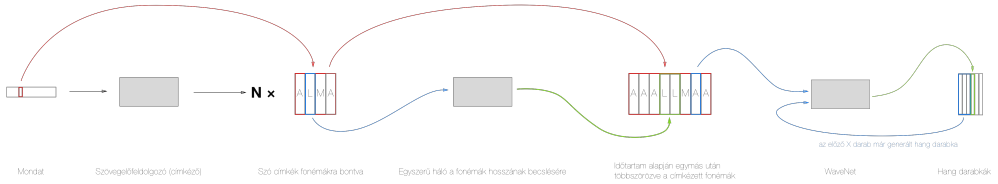
A Számítástechinka egyik meghatározó célja a számítógépekkel való emberszerű kommunikáció elérése. A Beszédszintézis ennek elengedhetetlen területe. Jelen dokumentum a szerzők a Mély Neurális Hálók nyújtotta lehetőségek a témában való alkalazásából szerzett tapasztalatainak összefoglalója.

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

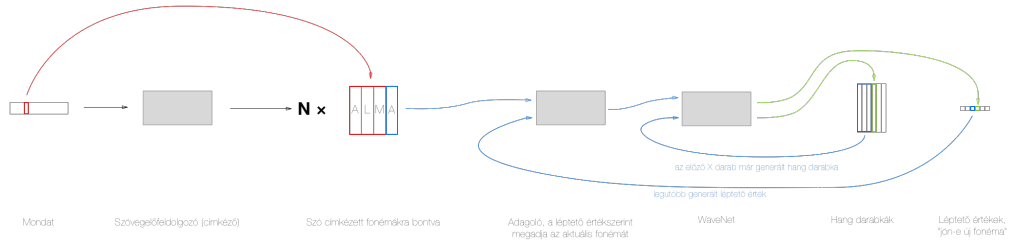
1 WaveNet tapasztalatok

Első célnak a beszédszintézis WaveNet[1] alapú megoldását tűztük ki. Ennek lényege az aktuális időpillanatban a hanghullám értékének az előző, már meghatározott értékek és a szövegfeldolgozásból kapott címkék segítségével történő meghatározása. Két rendszertervet dolgoztunk ki, azzal a különbséggel hogy a fonéma hosszának a becslése egy külön hálóban történik vagy a kimenetről van visszavezetve. Utóbbi esetben a hullámérték mellett lenne egy másik kimeneti érték is, amely azt adná meg hogy kezdődjön-e az új fonéma.

WaveNet külön fonéma becsült hálóval



WaveNet fonéma becsléssel kiegészítve



A megvalósítás során először már meglévő, GitHub-on elérhető implementációkat próbáltunk ki. Ezek közül az egyiket sikerült egy adott mondatra tanítanunk, azt vissza tudta generálni pontosan. Azonban más mondatokkal való továbbtanítás esetén már nem volt működőképes.

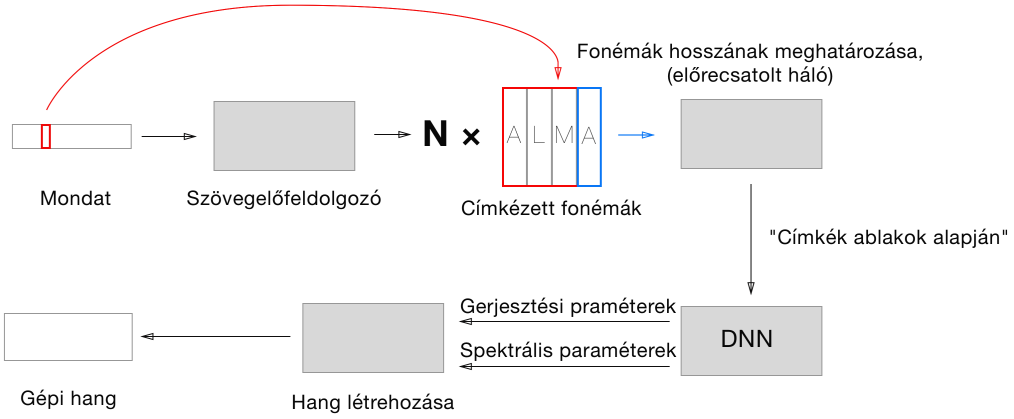
Ezután saját implementációval próbálkoztunk a DeepMind által publikált cikk alapján. Ez az implementáció lényegesen egyszerűbb volt mint a cikkben meghatározott, célunk csak valamilyen kis zörej előállítása volt. Azonban konstans (néma) hangon kívül mást nem sikerült előállítanunk.

A fent említett kísérletek után egyértelművé vált, hogy túl nagy feladatba kezdtünk bele. Ezért visszaléptünk az eredeti célunktól és folytatásképpen a hagyományos DNN alapú beszédszintézissel haladtunk tovább.

2 Egyszerű DNN model felépítése

DNN alapú rendszer esetén is az első fázis a szövegből fonéma alapú címkék előállítása, ezekről részletesebben következő fejezetben írtunk. Ezután meg kell határozni a fonémáknak a hosszát, erre egy egyszerű előrecsatolt háló alkalmas. Az idő paraméterekkel ellátott fonémákból becsülhetőek az aktuális időegységre jellemző gerjesztési és spektrális paraméterek. Ezekből a paraméterekből állítható elő az eredetihez hasonló gépi hang.

DNN rendszerl



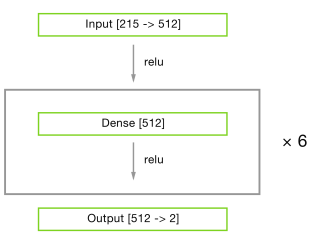
2.1 Háló modellek

Külön hálón tanítottuk a spektrális és a gerjesztési paramétereket.

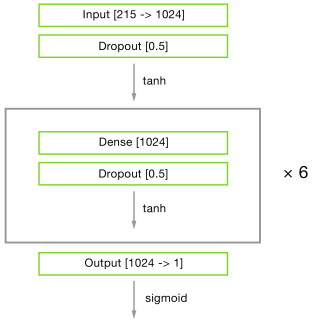
Gerjesztési paraméter problémáját két hálóra bontottuk szét. Az egyik a hang zöngésségét határozta meg a másik pedig az alapfrekvenciát, a végleges eredményeket a két háló által jósolt értékekből számítottuk ki. (Gyakorlatilag egy engedélyező jelként értelmeztünk a zöngésség értékét az alapfrekvencián.) A zöngésség meghatározása egy egyszerűbb probléma erre egy 6 rejtett réteget, rétegenként 512 neuront tartalmazó, RELU aktivációs függvényt, SGD optimalizálást és MSE költségfüggvényt alkalmazó hálót hoztunk létre. Az alapfrekvencia becslésére 6 rejtett réteget, rétegenként 1024 neuront tartalmazó, tanh és sigmoid aktivációs függvényt, SGD optimalizációt és MSE költségfüggvényt használó hálót alkalmaztunk. Utóbbi esetén a rétegenként alkalmaztunk Dropout-ot, tanítás során pedig early stopping-ot.

A gerjesztési paramétereket meghatározó háló hasonlóan épült fel az alapfrekvenciát becslő hálóhoz, de mivel az itt elért eredményeink nem elég jó minőségűek ezen még valószínűleg változtatnunk kell.

Zöngésség meghatározása



Alapfrekvencia meghatározása



3 Bemeneti és kimeneti paraméterek

3.1 Szövegfüggő címkék

A címke generálás egységeként egy mondatot használtunk fel. Ezt az nltk eszköz segítségével elemeztük majd az eredményekből állítottuk össze a címkéket. A címkék alapjául az aktuális fonémák szolgátak, később még ezek lettek továbbbontva időkeret szerint.A fonéma azonosítására OneHot kódolást alkalmaztunk, azaz minden fonémát 40 értékkel azonosítottunk.

A következő szöveg függő címkéket hoztuk létre:

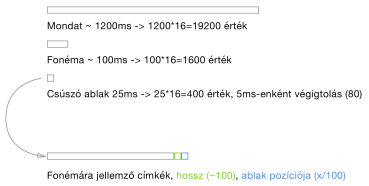
- aktuális fonéma és az őt körülvevő két-két fonéma
- hangsúly
- megelőző és követő fonémák száma a szóban
- távolság hangsúlyos fonémától mindkét irányban
- szó szófaja
- szó pozíciója a mondatban
- fonémák száma az aktuális szóban és két szomszédjában
- szavak száma a mondatba
- fonémák száma a mondatban

3.2 Bemeneti paraméterek

A szövegfüggő fonéma címkéket egészítettük ki a fonémában található időkeretek számával és a keret fonémán belüli elhelyezkedésével. Így a kerethez tartozó 215 bemeneti paraméter az alábbiakból tevődik össze:

- 0-200 A 40-40 paraméter a kvinfón(2-1-2) fonémáira
- 200-213 további szövegfüggő címkék
- 213 A fonémán belüli időkeretek száma
- 214 A keret elhelyezkedése a fonémán belül

Címkék létrehozásának folyamata



3.3 Kimeneti paraméterek

Az előző felsorolást folytatva az aktuális időkerethez tartozó elvárt kimenet paraméterek a következők:

- 215 a keret hangmagasság értéke
- 216-242 a Mel-Cepstrum 26 paramétere
- 242 a fonémán belüli keretek száma

3.4 Spektrális és gerjesztési paraméterek

Mint említettük a prediktálás alapja a spektrális és gerjesztési paraméterek megadása keretenként. Ezen paraméterek segítségével a PyPSTK [2] python csomag használatával állíthatjuk elő az audio kimenetet, valamint hasonlóképpen ezt a csomagot használjuk az adataink a tiszta hangból való előállítására.

3.4.1 Mel-Cepstrum

A paraméterek előállításához, visszafejtéséhez a PyPSTK Mel-Cepstrum alapú algoritmusát alkalmazzuk. Az algoritmust T. Fukada és társai fejlesztették ki. [3]

Ezzel a metódussal természetesen adatvesztést kapunk, azonban az így előállított hang még megfelel a mércéinknek. Viszont az általunk generált kimenetek értékelésénél ezt figyelembe kell vennünk.

4 Megvalósítás

4.1 Használt erőforrások

A háló összerakást saját gépen kezdtük el. Majd a véglegesítését és a tanításokat AWS szerveren, NVIDIA Tesla K80 12GB videokártyán végeztük. Jupyter notebook-ban futtatuk a tanításokat, ezek elmentett adatait aztán Tensorboard segítségével elemeztük.

4.2 Fejlesztés

A fejlesztés során scriptet írtunk az adatok generálására(*generate_dataset.py*). Ez állítja elő a tiszta adatfájlokból a hálók bemeneteit.

A különböző hálókat különböző jupyter notebookokban teszteltük. Ezek a *model* kezdetű notebookokban találhatóak.

A végső eredményeket a *results* notebookban összegeztük, ez végigfuttatható demonstrációként hamar előáll.

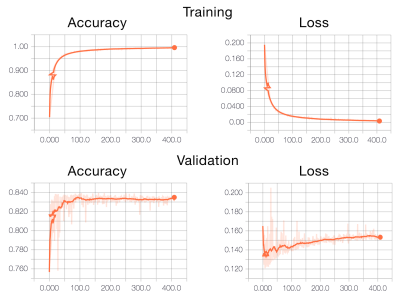
4.3 Eredmények

4.3.1 Zöngésség

Zöngésség becslésére nagyon jó, alapfrekvencia (pitch érték) meghatározására használhatóan jó és a spektrális paraméter (Mel-Cepstrum) jósolására kezdetleges eredményeket sikerült elérnünk. (utóbbiról ezért nem is mellékeltünk tanítási és validációs grafikonokat). Hiperparaméter optimalizálás tekintetében egyelőre kézzel dolgoztunk (ez még javításra szorul), már így is sikerült használható pontosságot elérnünk.

(A grafikonk y tengelyén az MSE hiba értéke x tengelyén az elvégzett epoch-ok száma látható.)

Zöngésség becslése egyszerűbb, futásidő tekintetében gyorsabb feladatnak bizonyult. Ezért itt nem alkalmaztunk early stopping-ot, hanem a több próbálkozás után a fixen 410 epoch-kal történő tanításnál maradtunk.
A teszt adatainkon elért eredmények:
pontosság 0.8342
költségfüggvény: 0.1560
(MSE hiba értékekkel számítva)



4.3.2 Alapfrekvencia

Az alapfrekvencia tanítása esetén alkalmaztunk early stoppig-ot. Ez azonban (a paramétereinek állítása után is) túl hamar állt meg. Ennek következtében több tanítást is elvégeztünk egymás után. Az itt kapott eredményeink messze nem olyan szépek mint a zöngésségé, de ezek is használhatónak bizonyultak.

A teszt adatainkon elért eredmények:

pontosság: 0.4243

költségfüggvény: 0.07214

(MSE hiba értékekkel számítva)

4.3.3 MC paraméterek

MC tanításakor kiütközött, hogy a bemeneti paramétereink nagyon megegyezőek voltak fonémán belül, így darabos lett a tanítás. Ez a magasabb MC értékekre jelentős hibát okozott, így megpróbáltuk kevesebb mc érték használatát pontosítani.

A teszt adatainkon elért eredmények:
pontosság 0.2927
költségfüggvény: 0.2086
(MSE hiba értékekkel számítva)

4.3.4 Fonéma hossz

Fonéma hossz becslésnél csoportosítást valósítottunk meg, a fonémákat 8 csoportba osztottuk, a csoportok között 12,5 ms eltéréssel és a csoportra végeztünk classificationt. Erre azért volt szükség mivel viszonylag kevés, géppel annotált tanító adat állt a rendelkezésünkre, ezek alapján nem sikerült használható eredményeket elérni. Az adatokban szereplő eredeti hosszak vizsgálata után ezt a csoportosítási módot választottuk, két szempont a megfelelő differenciálódás és a jelentősen kilógó adatok levágása volt. Tehát minden fonémánk 40 és 140 ms közé került. (A tanító adatainkat is ez alapján módosítottuk még a feldolgozási fázisban, minden keretnél hosszabb fonémát 140 ms-nél levágunk.)

A teszt adatainkon elért eredmények:
pontosság 0.4619
költségfüggvény: 0.02419
(MSE hiba értékekkel számítva)

4.3.5 Teljes becslő

A fenti paraméterek ismeretében teljes TTS végezhető.

A mondat alapján fonéma hosszakat becslünk, majd párhuzamosan zöngésség-alapfrekvencia és mc paraméter becslés folyik, végül a kapott eredményekből előállítható a kimenet.

5 További lehetőségek

Jelenleg van egy kész rendszerünk, amely bevitt szöveg alapján hangot állít elő. Minőségi problémák azonban vannak még az előállított hanggal. Ezen kétféleképpen tervezünk még javítani egyrészt további új adatokon való tanítással, valamint az előfeldolgozási folyamatunk javításával (több, kifejezőbb címke bevezetése). Ezen felül további minőségi javulást lehet elérni több spektrális paraméter alkalmazásával is. Minőség javítás mellett még további cél saját felhasználói felülettel rendelkező rendszer létrehozása. Itt többféle út is lehetséges egyrészt szerver kliens jellegű felépítés másrészt erőforrásigénytől függően egyedülálló rendszer.

A fenti bekezdés pontokban:

- minőség javítás új adatokkal
- címkézés felülvizsgálata, további spektrélis paraméterek
- felhasználói felület fejlesztése

Hosszútávú célként említhető még a WaveNet jellegű implementáció további fejlesztése.

Irodalom

[1] van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., ... & Kavukcuoglu, K. (2016).*WaveNet: A generative model for raw audio.* arXiv preprint arXiv:1609.03499.

[2] SPTK Working Group. (2009) *Speech signal processing toolkit (SPTK).* http://sp-tk.sourceforge.net

[3] Fukada, T., Tokuda, K., Kobayashi, T. & S. Imai, (1992) *An adaptive algorithm for mel-cepstral analysis of speech* Proc. ICASSP-92 , pp.137–140, Mar. 1992.

[4] Qian, Y., Fan, Y., Hu, W., & Soong, F. K. (2014, May). *On the training aspects of deep neural network (DNN) for parametric TTS synthesis.* In 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 3829-3833). IEEE.

[5] Chollet, F. (2015). *Keras*. GitHub repository: https://github.com/fchollet/keras.

[6] Bird, S. (2006, July). *NLTK: the natural language toolkit.* In Proceedings of the COLING/ACL on Interactive presentation sessions (pp. 69-72). Association for Computational Linguistics.

[7] TensorBoard. Available at
http://www.tensorflow.org/versions/r0.7/how_tos/summaries_and_tensorboard/index.html Accessed 25 April 2016.

Köszönetnyílvánítás

A dokumentum és a szerzők munkái nagyban támaszkodnak a Budapesti Műszaki és Gazdaságtudományi Egyetemen tartott Deep Learning a gyakorlatban Python és LUA alapokon tárgy keretei között elhangzott információkra