MicroProfile

Optimizing Enterprise Java for a microservices architecture

Björn Beha und Suhay Sevinc

***Zusammenfassung*—This document is a model and instructions for LaTeX. This and the IEEEtran.cls file define the components of your paper [title, text, heads, etc.]. *CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract.**

I. Einführung

DEr Hype um den Modularisierungsansatz Microservices dauert an.r

II. Multilinguale Spracherkennung

Die Kernidee der mehrsprachigen Spracherkennung ist bei den verschiedenen Architekturen dieselbe. Die Hidden-Layer des Deep Neural Networks können als ein intelligentes Merkmalsextraktionsmodul betrachtet werden, welches aus mehreren Quellsprachen trainiert wird. Nur die Ausgabeschicht liefert eine direkte Übereinstimmung mit den relevanten Klassen. So lassen sich die Extraktoren für eine Reihe verschiedener Sprachen gemeinsam nutzen. Wie im vorherigen Kapitel bereits erläutert wurde, lässt sich somit besonders das Problem beim Lernen der tiefen neuronalen Netze entgegenwirken. Diese lassen sich sind aufgrund ihrer Parameter und dem sogenannten Backpropagation-Algorithmus langsamer trainieren als andere Modelle. Ein weiterer Vorteil, den dieser Ansatz bietet ist, dass auch mit Sprachen, die nur einen geringen Satz an markierten Trainingsdaten bietet, erlernt werden können, indem Elemente anderer Sprachen übertragen werden. Merkmale, die aus diesen neuronalen Netzen extrahiert werden, lassen sich kombinieren, um so die Erkennungsgenauigkeit zu verbessern [1]. Eine gemeinsame Nutzung wird ermöglicht, indem Phoneme gemeinsam genutzt werden. Phoneme sind als kleinste, bedeutungsunterscheidende Einheiten der Lautsprache definiert. Phoneme werden zur Repräsentation der Aussprache genutzt. Um obige Ansätze zu nutzen, müssen Beziehungen zwischen den akustischen Signalen der Sprachen erkannt werden. Jede Sprache besitzt dabei ihre eigenen Charakteristika. In der Sprachübergreifenden Erkennung gibt es einen Satz aus trainierten sowie untrainierten bzw. schlecht trainierten Phonemen, die erkannt werden müssen. Die Töne einer Sprache müssen mit einem ähnlichen bzw. dem ähnlichsten trainierten Ton einer anderen Sprache ersetzt werden. Beispielsweise gibt es den Ton /y/, welcher im Wort ‚süß' vorkommt. Wenn ein System nun mit der deutschen Sprache genutzt wird, welches nur in anderen Sprachen trainiert wurde, muss der ähnlichste Sound zu /y/ gefunden werden [5]. So lassen sich phonetische Klassen aus mehreren Sprachen unterscheiden. Ein Transfer des Modells ist trivial. Es wird leidglich eine neue Softmax-Schicht angelegt und trainiert. Die Softmax-Funktion wird zur Klassifikation verwendet [6]. Die Ausgabenkoten dieser Schicht entsprechen dann den Senonen der Zielsprache. Senonen beschreiben lediglich das Betrachten des lautlichen Kontextes der einzelnen Phoneme und stellen gebundene Triphonzustände dar [1]. Die Kontexte können komplex sein [7]. Die Softmax-Schicht wird nur auf die entsprechende Sprache trainiert. Weitere Verbesserungen lassen sich erzielen, indem das gesamte Netzwerk zusätzlich auf die neue Sprache abgestimmt wird. Ein solche Architektur ist in Abbildung (...) illustriert. Sie zeigt die gemeinsam genutzten Schichten, die die Merkmale extrahieren sowie die unterschiedlichen Input-Datensätze. Jede Sprache hat ihre eigene Softmax-Ebene. Mit der Softmax-Funktion lassen sich hier die Zustände des akustischen Modelles vorhersagen bzw. entsprechende Wahrscheinlichkeiten schätzen. Wird ein neuer Datensatz in das System gegeben, werden nur die sprachspezifische Schicht sowie die Hidden Layer angepasst. Andere Softmax-Schichten bleiben intakt. Nach dem trainieren der fünf Sprachen ist das System in der Lage diese fünf Sprachen zu erkennen. Die Erweiterung um eine Sprache ist trivial. Kommt eine weitere Sprache hinzu, wird leidglich eine neue Softmax-Ebene an das vorhandene Netzwerk angefügt und trainiert [1].

Ein Vergleich eines monolingualen Deep Neural Networks und eines multilingualen Deep Neural Networks ist in Tabelle (...) aufgeführt. Das monolinguale Netzwerk wurde hierbei nur mit der entsprechenden Sprache trainiert, während das multilinguale System mit allen vier Sprachen trainiert wurde. Dabei wird die prozentuale Wortfehlerrate (Word error rate, WER) angegeben. Es ist zu erkennen, dass das multilinguale System das monolinguale in allen Sprachen übertrifft. Diese Verbesserung ist dem sprachübergreifenden Wissen zuzuschreiben [1].

In [2] wurden eine Reihe weiterer Versuche durchgeführt, um die Wirksamkeit eines solchen Systems zu evaluieren. Dabei wurde zwei verschiedene Zielsprachen verwendet. Zum einen das amerikanische Englisch, welches phonetisch nahe an den europäischen Sprachen der Tabelle (...) liegt und Mandarin-Chinesisch, welches weit von den europäischen Sprachen entfernt ist. Die tatsächliche Erkennung der Sprache ist dabei trivial. Die Schallwellen, die beim Sprechen produziert werden, lassen sich über einen elektroakustischen Wandler (Mikrophon) in ein elektrisches Signal umwandeln. Dieses elektrische Tonsignal wird daraufhin in Zahlen bzw. Bits konvertiert (sampling) und über Vorverarbeitung entsprechend aufbereitet, um es in ein neuronales Netz zu speisen [4]. Beim genauen Vorhersagen des gesprochenen kommt die Sprachidentifikation ins Spiel, durch welche Wörter ausgeschlos-

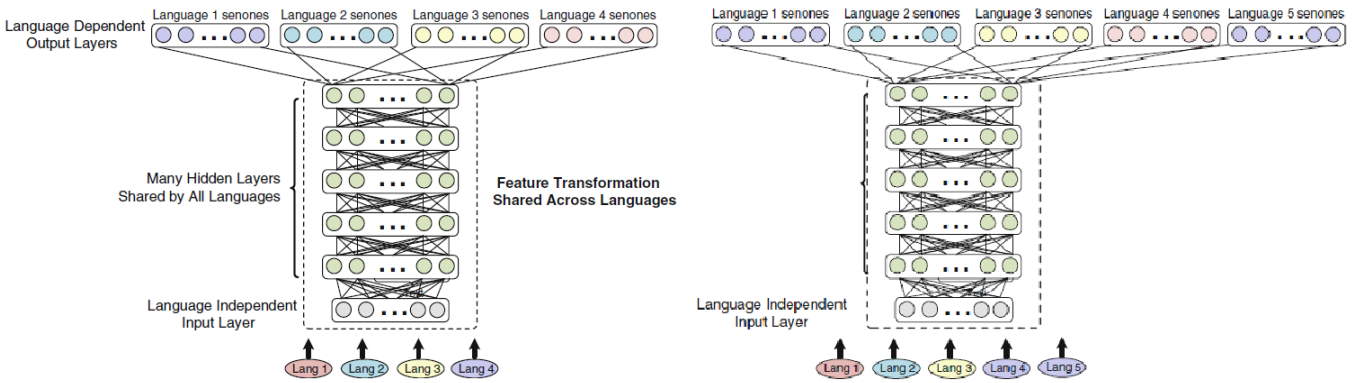

Abbildung 1. Hinzufügen einer neuen Sprache [1]

	FRA	DEU	ESP	ITA
Test set size (words)	40k	37k	18k	31k
Monolingual DNN WER	28.1%	24.0%	30.6%	24.3%
Mulitlingual DNN WER	27.1% (-3.6%)	22.7 (-5.4%)	29.4 (-3.9%)	23.5% (-3.3%)

Tabelle I
RELATIVE WORTFEHLERRATE

sen werden, die ebenfalls in Frage kommen, allerdings zum Wortschatz einer anderen Sprache gehören. Hat ein System z.B. die deutsche Sprache erkannt, wird für das Wort „Hello“ immer „Hallo“ vorhergesagt, da dies naheliegender ist. Es wird hier mit statistischen Modellen gearbeitet, um anzugeben mit welcher Wahrscheinlichkeit welches Wort vorkommt oder aufeinander folgen können. Dabei gibt es verschiedene Lösungsansätze, um das gesprochene Vorherzusagen. Oft werden tiefe neuronale Netze in Verbindung mit Hidden Markov-Modellen eingesetzt. Diese hybriden Systeme werden in der Literatur oft untersucht und beschrieben. Ein allerdings leistungsfähigeres Modell bieten die Recurrent Neural Networks. Diese Form von neuronalen Netzen werden heutzutage eingesetzt und erreichen hohe Genauigkeiten [1].

III. RECURRENT NEURAL NETWORKS

Im Bereich der Spracherkennung werden heutzutage sogenannte Recurrent Neural Networks eingesetzt, durch welche die Netzwerke ihre Spracherkennungsgenauigkeit erreichen. Das Modell dieser Netze erlaubt gerichtete zyklische Verbindungen zwischen den Neuronen, wodurch es mit einem temporalen Verhalten ausgestattet wird. Recurrent Networks sind somit ideal zum Lernen von Datensequenzen geeignet. Sprache, also kontinuierliche Audiostreams fallen somit ebenfalls in das Anwendungsgebiet dieser Netzwerke. Diese Form von neuronalen Netzen unterscheidet sich grundlegend von dem Feed-Forward-DNN, da es nicht nur basierend auf Eingaben arbeitet, sondern auch auf interne Zustände zurückgreift. Diese internen Zustände speichern die vergangenen Informationen in der zeitlichen Reihenfolge, in welcher diese verarbeitet wurden. Somit ist ein RNN deutlich dynamischer, als ein Deep Neural Network, welches lediglich eine statische Eingabe-Ausgabe-Transformation durchführt. Dabei wird eine Erweiterung des Backpropagation-Algorithmus eingesetzt. Die Backpropagation-Through-Time-Methode sorgt für das Berechnen der Gradienten. Diese werden im Gegensatz zum Standard-Algorithmus über die einzelnen Zeitschritte aufsummiert. In dieser Erweiterung des Backpropagation, welche in Recurren Neural Networks eingesetzt wird, werden lediglich die Parameter einzelnen Zeitschritte zwischen den Ebenen geteilt. In Abbildung (...) ist ein vereinfachtes Modell illustriert [1].

Die Abbildung zeigt eine Folge von Iterationen. Der Input ist in obiger Darstellung x, s bezeichnet den Schritt und E den Hidden State, welcher sich beim Eingeben des Inputs ergibt. Ein Recurrent Network gibt somit nicht nur den Input an die nächste Iteration, sondern Input sowie den resultierenden Zustand E. Somit beeinflussen die vorhergehenden Schritte die folgenden. Dies führt zu einem Problem -dem Verschwinden von Information bzw. dem Vanishing Gradient Problem, welches sich dadurch ergibt, dass RNNs nicht in der Lage sind, auf Informationen zurückzugreifen, die weit in der Vergangenheit liegen. Wenn eine Datensequenz lange ist und das System versucht die gesagten Worte vorherzusagen, kann es sein, dass der Kontext bereits vergessen wurde und eine inkorrekte Vorhersage stattfindet. Somit kommt eine erweiterte Form des RNNs zum Einsatz. Dieses wird Long-Short-Term-Memory (LSTM) genannt und erzielt enorm gute Ergebnisse in automatischen Spracherkennungssystemen [1][3]. Diese Netzwerke sind somit in der Lage anhand des Kontextes zukünftige Wörter vorherzusagen und so ihre Genauigkeit zu erhöhen. Auch mit verrauschten Aufnahmen oder schlechteren Bedingungen beim Aufnehmen des Gesprochenen kann diese Form von Netzwerken bessere Ergebnisse erzielen. Aufgrund dessen wurden LSTM-Netzwerke entwickelt, die zur Lösung des Problems beitragen. Dabei werden Recurrent Neural Networks mit einer Speicherstruktur erweitert, was zur namensgebenden Lang-Kurzzeit-Speicherung führt. Diese erlauben die

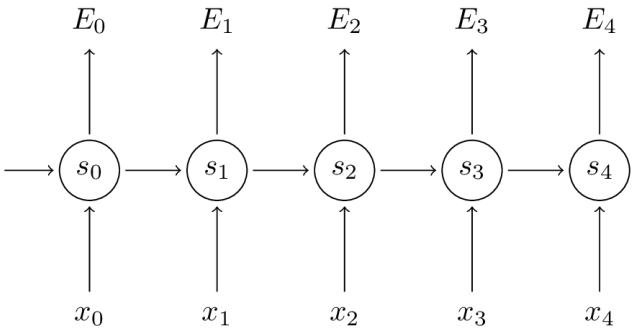

Abbildung 2. Modell des Recurrent Neural Network [1]

Erkennung zeitlich ausgedehnter Muster und das Erkennen von Zusammenhängen von zeitlich getrennten Ereignissen. Somit eignen sich die Netzwerke um Zeitreihen zu verarbeiten und vorherzusagen. Sogar, wenn zwischen wichtigen Ereignissen sehr lange Verzögerungen liegen, die eine unbekannte Länge aufweisen. Auch bei der Erlernung geräuschverzerrter und hallender Sprachmerkmale kann dieses Modell genutzt werden [1]. Die grundsätzliche Idee dabei ist es über elementweise Multiplikationen den Informationsfluss in dem Netzwerk zu steuern. Es kann als komplexe und intelligente Netwerkeinheit betrachtet werden, welche Informationen über einen langen Zeitraum speichern kann. Dies wird durch die Gating-Struktur erreicht, die bestimmt, wann die Eingabe signifikant genug ist, um sich daran zu erinnern, wann sie sich die Information weiter merken oder vergessen sollte und wann sie die Information ausgeben sollte. Dies geschieht über verschiedene Gates innerhalb einer LSTM-Zelle (...). Ein Gate ist dabei nichts weiter, als eine Reihe von Multiplikationen bzw. Matrixoperationen [1]. Das System ist somit in der Lage aus dem Kontext heraus genaue Vorhersagen zu treffen, wodurch Spracheerkennung deutlich präziser wird. Allerdings ist es selbst heute nicht möglich das Spracherkennungsproblem allgemein zu lösen. Spracherkennungssysteme werden somit nur für bestimmte Anwendungsfälle oder Szenarien konzipiert. Mit einer solchen Spezialisierung auf entsprechende Anwendungsgebiete können zum einen höhere Genauigkeiten erreicht werden und zum anderen wird nicht so viel Rechenleistung und Speicher benötigt [4]. Vor allem bei der multilingualen Spracherkennung besteht die Schwierigkeit Gemeinsamkeiten verschiedener Sprachen zu nutzen, um Sprachen mit wenig Trainingsdaten mit einer ausreichenden Genauigkeit anzubieten. Es gilt die Sprachen zu finden, die zur besten Erkennungsleistung der neuen Sprache führen. Dabei müssen Beziehungen zwischen den Sprachen erkannt werden. Problematisch ist auch, dass gleiche Phoneme je nach Sprecher, Sprache etc. variieren, was dazu führt, dass Phoneme nur im Kontext betrachtet werden (Triphone).

IV. Trainingsvorgang

A. Trainingsvorgang

Der Trainingsvorgang basiert auf ein vollständig verbundenes mehrschichtiges Deep Learning-Netzwerk mit einem Feed-Forward-Prinzip. Hier existieren die Schichten:

Input-Schicht
Hidden-Schicht
Output-Schicht

Die Input-Schicht stellt dabei die Eingangsdaten dar, welche als Trainingsmaterial für den Trainingsvorgang verwendet werden. Bei diesen Daten handelt es sich um Sprachaufnahmen. Bei Bedarf können diese Sprachaufnahmen dementsprechend vor-verarbeitet werden, wie zum Beispiel durch Einsatz von Filtern. Anschließend können die Daten in die Netztopologie eingespeist werden. In der Hidden-Schicht geschieht das eigentlich Training, welches normalerweise durch die Sigmoid-Funktion aktiviert wird [2].

$$sigm(x) := \frac{1}{1 + e^{-x}}$$

Die Ableitung dieser Funktion gibt Hinweise, die wiederum Nachteile sich bringt.

$$sigm(x)' := \frac{e^x}{(e^x + 1)^2}$$

Hierbei werden die Schichten durchlaufen, welches das Training darstellt. Beim Backpropagation-Algorithmus wird das Netz rückwärts berechnet. Hierzu wird die Ableitung der Sigmoid-Funktion benötigt. Dieses Verfahren wird benötigt, um fehlerhafte Kantengewichte zu finden und anzupassen. Bei dieser Anpassung Hier kann ein Lernverlust bis zu maximal 25 Prozent entstehen, welches das Maxima von $sigm(x)'$ repräsentiert. Da bei der Sprachverarbeitung eine große Datenmenge anfallen, könnte der Trainingsverlust zu groß sein. Somit wäre das Lernen nicht mehr effizient, welches in 3 dargestellt ist [3]. Anstelle der Sigmoid-Funktion wird in den modernsten Deep-Learning-Netzen Rectified linear Units ($ReLUs$) verwendet. Dabei gibt diese Funktion bei einem negativen Input-Wert, den Wert null zurück. Bei einem positiven Input-Wert wird dieser als Output-Wert übernommen.

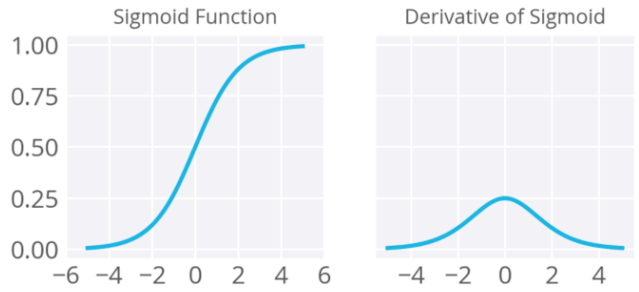

Abbildung 3. Darstellung der Sigmund-Freud-Funktion und dessen Ableitung [3]

Diese Funktion ist menschlichen Neuronen am ähnlichsten und bringt zudem eine effizienter Verarbeitungsgeschwindigkeit [4]. Hinzukommt, dass in die Formel noch die Kantengewichtungen der Neuronen benötigt werden. Aktivierungsfunktion x_j über den Index j gemapped. Die meisten Frameworks wie TensorFlow und TFLearn machen es einfach, ReLUs auf Hidden-Schicht anzuwenden, sodass diese Implementierungen bereits zur Verfügung gestellt werden. [1].

$$y_j = ReLU(x_j) = max(0, x_j)$$
$$x_j = b_j + \sum x_{ij} * y_j \quad (1)$$

Als nächstes folgt die Output-Schicht, welches die Eingangsdaten dann zu den Klassen (Vorhersagen) zuordnet. Diese Schicht ist als Softlayer konfiguriert, welches die Klassen in eine eindimensionale Matrix kategorisiert. Eine Klasse steht für eine Sprache, die gelernt werden soll. Dabei ist die Matrix in dem Zahlenintervall $[0, 1]$ normalisiert. Die endgültige Sprachidentifikation geschieht über normalisierten Werte, welches in Abbildung 4 dargestellt werden. Diese können in Wahrscheinlichkeits-Werte umgerechnet werden [3]. Die

Abbildung 4. Klassenzuordnung über Wahrscheinlichkeiten in der Softmax-Konfiguration [3]

Vorhersagen der Output-Schicht geschieht durch die Funktion $p(j)$. Dabei steht der Index l für die jeweilige Klasse, also die Sprache, die gelernt werden soll.

$$p(j) := \frac{exp(x_j)}{\sum_l exp(x_l)}$$

Für den vorhin erwähnten Backpropagation-Algorithmus wird ebenfalls eine Kostenfunktion benötigt, welche für Korrektur der Kantengewichte ist. Diese geschieht durch Cross-Entropy-Loss-Funktion.

$$C := \sum_l t_j * log(p_j)$$

Diese die Funktion misst die Abweichungen der Kantengewichte der Netztopologie und passt diese rückwirkend an. Der Cross-Entropy-Verlust nimmt zu, wenn der vorhergesagte Wert von der tatsächlichen Beschriftung abweicht [5]. Bei t_j handelt es sich um die Klasse, für die der Verlust berechnet wird [1].

B. Netztopolgie

Die Netztopologie beschreibt die Infrastruktur des Netzes. Die Auswahl der Topologie bestimmt die Qualität des Trainingsvorgangs. Aufgrund dessen fallen Topologien von Ansatz zu Ansatz unterschiedlich aus, welche auch unterschiedliche Spracherkennungsresultate liefern. Dafür wird die Topologienvorschlag von Gonzales et al. betrachtet. Für die Eingangsdaten werden 40 Filterbanken verwendet. Diese werden benötigt, um die Daten samplen zu können. Dadurch entsteht eine Menge 26 Knoten innerhalb der Input-Schicht. Um unerwünschte Latenzzeiten der Frames zu vermeiden wird ein asymmetrischer Kontext verwendet. Die Hidden-Schichte beträgt hier vier Ebenen mit einer Gesamtzahl von 2560 Units. Die Output-Schicht enthält wie bereits erwähnt eine Softmax-Konfiguration, welche eine Dimension der Anzahl der Zielsprachen besitzt.

C. Verbesserung des Trainingsverfahrens durch Multitasking learning (MTL)

Bei Machine Learning wird besonders wert gelegt, bestimmte Metriken, wie beispielsweise Klassifizierungs-Genauigkeit und Trainingsdauer, zu optimieren. Daraufhin wird das Modell soweit optimiert bis die Leistung des Modells nicht mehr gesteigert werden kann [6]. Das Lernen der einzelnen Sprachen läuft sequenziell ab. Hier setzt das Multitasking Learning (MTL) ein. Hier werden mehrere Lernaufgaben gleichzeitig erledigt statt sequentiell, um das Trainingsvefahren effizienter zu gestalten. Das führt zu einer verbesserten Lerneffizienz und Vorhersagegenauigkeit. Im Klassifizierungskontext zielt MTL darauf ab, die Leistung mehrerer Klassifizierungsaufgaben zu verbessern, indem sie gemeinsam erlernt werden [7]. Ein Beispiel hierfür ist ein Spamfilter. Der Schlüssel zur erfolgreichen Anwendung von MTL besteht darin, dass die Aufgaben miteinander verknüpft werden können. Dies bedeutet nicht, dass die Aufgaben ähnlich sein müssen. Stattdessen bedeutet es, dass Aufgaben auf verschiedene Ebenen abstrahiert und geteilt werden. Wenn die Lernaufgaben tatsächlich ähnlich sind, können sie gemeinsam gelernt werden. Dabei kann das Wissen zwischen Aufgaben auf andere Lernaufgaben übertragen werden, welches die Trainingsdauer deutlich verkürzt. MTL ist vor allem dann nützlich, wenn die Größe des Trainingssatzes im Vergleich zur Modellgröße klein ist. Dabei wird grundsätzlich zwei Arten von MTL unterschieden: Hard parameter sharing und soft parameter sharing.

Hard parameter sharing stellt das meist genutzte Art dar [6]. Es wird normalerweise auf die hidden-Schicht angewendet, indem die Aufgaben gemeinsam gelernt werden, während die spezifischen Aufgaben seperat gelernt werden.

Dies wird in Abbildung 5 dargestellt. Dieser Ansatz reduziert das Risiko von Overfitting erheblich. Je mehr Aufgaben gleichzeitig gelernt wird, desto mehr muss das Modell eine Repräsentation finden, die alle Aufgaben erfassen muss. Dadurch ist die Chance des Overfittings deutlich geringer.

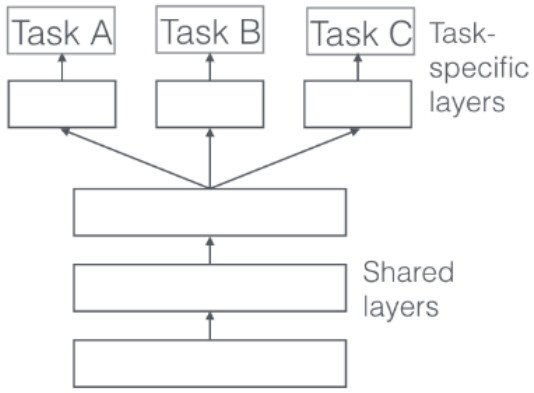

Abbildung 5. Hard parameter sharing auf die Hidden-Schicht angewendet [3].

V. Fazit

VI. Ausblick

Literatur

[1] J. Gonzalez-Dominguez, D. Eustis, I. Lopez-Moreno, A. Senior, F. Beaufays, and P. J. Moreno, "A real-time end-to-end multilingual speech recognition architecture," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 4, pp. 749–759, 2015.

[2] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[3] GitHub, "Kulbear/deep-learning-nano-foundation," 2017, (Accessed on 2018-04-15). [Online]. Available: https://github.com/Kulbear/deep-learning-nano-foundation/wiki/ReLU-and-Softmax-Activation-Functions

[4] M. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, and G. Hinton, "On rectified linear units for speech processing," in *38th International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, 2013.

[5] M. Cheatsheet, "Loss functions — ml cheatsheet documentation," 2017. [Online]. Available: http://ml-cheatsheet.readthedocs.io/en/latest/lossfunctions.html

[6] S. Ruder, "An overview of multi-task learning in deep neural networks," 2017. [Online]. Available: http://ruder.io/multi-task/

[7] Y. Lu, F. Lu, S. Sehgal, S. Gupta, J. Du, C. H. Tham, P. Green, and V. Wan, "Multitask learning in connectionist speech recognition," 2015.