VICTOR GIRALDI

ConvBlocks vs. Recurrent Layers: Optimalisatie van Neural Network Architecturen voor MIT-BIH Aritmie Classificatie

Naam: Victor Giraldi
Studentnummer: 1694420
Cursus: Master of Informatics - Deep Learning
Datum: 01-07-2025
GitHub Repository: https://github.com/vgiraldi99/MADS-ML-Heart.git

Abstract – In dit onderzoek wordt het vermogen van Convolutional Neural Networks (CNN's) en Recurrent Neural Networks (RNN's) voor ECG-aritmie classificatie op de MIT-BIH Arrhythmia Dataset met 109.446 samples verdeeld over 5 klassen getest. De hypothese stelt dat CNN's superieure patroonherkenning bieden ondanks het tijdsafhankelijke karakter van ECG-signalen. CNN-architecturen werden ontwikkeld met modulaire ConvBlocks en Multi-Head Self-Attention, waarbij ECG-signalen werden getransformeerd naar 2D-matrices van 16×12. RNN-varianten omvatten LSTM en GRU-architecturen met variabele hidden sizes. Ray Tune en HyperOpt optimaliseerden meer dan 300 configuraties, waarbij recall werd geprioriteerd. De resultaten tonen dat CNN's presteerden met maximale recall van 0.965 en accuracy van 0.969, behaald door een compacte architectuur met 64 hidden units, één convolutionele laag en LeakyReLU activatie. RNN's bereikten hoogste recall van 0.939 met GRU-architecturen van 168 hidden units. CNN's vertoonden echter meer overfitting (training loss: 0.058-0.091 vs test loss: 0.122-0.242), terwijl RNN's betere generalisatie demonstreerden. De hypothese wordt bevestigd dat CNN's effectiever zijn voor ECG-aritmie classificatie, met LeakyReLU als dominante activatiefunctie en GRU's als beste RNN-variant. Het onderzoek benadrukt het belang van balancering tussen prestatie en generalisatie in medische AI-toepassingen.

Inleiding

De complexiteit van de optimalisatie van deep learning modellen heeft vaak te maken met wat voor probleem er wordt onderzocht. In dit onderzoek wordt besproken hoe hyperparameter optimalisatie technieken zijn toegepast op een drietal modellen om de optimale parameters te vinden voor het ECG Arrhythmia classificatieprobleem.

In dit onderzoek zullen er moderne hyperparameter optimalisatie technieken worden gebruikt voor het vinden van de optimale parameters. Door het gebruiken van Ray samen met HyperOpt worden er in totaal meer dan 300 verschillende configuraties getest om zo de optimale architectuur samen met parameters te vinden. In dit experiment is onderzoek gedaan naar het toepassen van CNN en RNN-architectuur variaties.

Dataset

Voor het huidige onderzoek wordt de **MIT-BIH Arrhythmia Dataset**[1] gebruikt. Deze dataset heeft de volgende metadata:

Aantal samples:	109,446
Aantal categorieën:	5 klassen ['N': 0, 'S': 1, 'V': 2, 'F': 3, 'Q': 4]
Sample frequence:	125 Hz
Klassen Distributie:	'N': 82.77%, 'S': 2.54%, 'V': 6.61%, 'F': 0.73%, 'Q': 7.35%

Hypothese

In het huidige onderzoek worden CNN- en RNN-architecturen tegenover elkaar gezet, waarbij RNN-architecturen gebruikelijk zijn bij Machine Learning problemen met een tijdsfactor, omdat RNN-architecturen profiteren van hiddenstates en gates die een vorm van selectief-geheugen bieden.

Convolutional Neural Networks (CNN's) werken door het convuleren van waardes in een 2-dimensionale matrix door het gebruiken van reading kernels. Hierdoor zijn CNN's beter geschikt in het herkennen van patronen in twee dimensies. Maar door het omzetten van de ECG naar een twee-dimensionale weergave is het mogelijk CNN's te gebruiken voor het huidige probleem.

Hoewel bij problemen met tijdsgebaseerde afhankelijkheden vaak de voorkeur uitgaat naar Recurrent Neural Networks (RNNs), wordt in dit onderzoek verwacht dat Convolutional Neural Networks (CNN's) effectiever zullen zijn voor het huidigde classificatie probleem. De verwachting is dat de sterke capaciteit van CNN's op het gebied van patroonherkenning een doorslaggevend voordeel zullen bieden bij het onderscheiden van de verschillende aandoeningen binnen de dataset.

Methodologie

Datapreparatie

Voor de huidige dataset is een voorgedefinieerde dataloader geleverd die alle ECG-signalen middels zero-padding normaliseert naar een uniforme lengte van 192 samples. Deze preprocessing is noodzakelijk omdat de originele MIT-BIH signalen variabele lengtes hebben. Voor CNN-architecturen wordt het eendimensionale signaal getransformeerd naar een tweedimensionale matrix van dimensies (16, 12).

Architectuurontwerp CNN

De voorgestelde CNN-architectuur is modulair opgebouwd en bestaat uit een opeenvolging van Convolutionele Blokken (ConvBlocks), gecombineerd met periodieke MaxPooling-lagen en optioneel een Multi-Head Self-Attention (MHSA) mechanisme. Deze attention-layer is geïnspireerd van het onderzoek "*CNNtention: Can CNNs do better with Attention?*" (Kapila, Glattki, & Rathi, 2024).

Voor de convolutielagen wordt gebruikgemaakt van kernels met een afmeting van 3×3 en padding van 1, gericht op het behouden van ruimtelijke resolutie tijdens de feature-extractie. Het aantal ConvBlocks wordt behandeld als een hyperparameter binnen het hyperparameter- optimalisatieproces. Na elke twee

[1] https://physionet.org/content/mitdb/1.0.0/

ConvBlocks wordt een MaxPooling-operatie toegepast, gevolgd door een optioneel MHSA-mechanisme om de capaciteit voor patroonherkenning te vergroten.

De dense-layer van het model bestaan uit lineaire functies samen met een activatie functie. Binnen deze dense laag wordt geëxperimenteerd met verschillende activatiefuncties (zoals ReLU, GELU en Leaky ReLU), die als hyperparameter wordt meegenomen.

Architectuurontwerp RNN

De RNN-architectuur behandeld de ECG-data als sequentiële data waarbij patronen worden gemodelleerd door meerdere (recurrent) lagen. De backbone van het model bestaat in dit onderzoek van GRU of LSTM lagen met variabele diepten en hidden-sizes die als hyperparameters worden geoptimaliseerd.

PyTorch's MultiheadAttention wordt toegepast op de RNN-outputs om attention weights te berekenen over de gehele sequentie. Dit stelt het model in staat om te focussen op relevante tijdsperiodes ongeacht hun positie in het signaal.

Hyperparameter Framework

Voor de systematische optimalisatie van beide architecturen werd Ray Tune geïmplementeerd met een HyperOpt search algoritme. HyperOpt biedt een intelligente parameter exploratie omdat er wordt gezocht rondom de instellingen van voorgaande trails die goed hebben gepresteerd volgens de aangegeven metriek. Daarbij maakt HyperOpt het mogelijk om niet veelbelovende parameters vervroegt te stoppen.

Omdat in het huidige onderzoek een model wordt getraind dat een bepaalde ziekte moet herkennen samen met dat het een ongebalanceerde dataset is zijn er twee belangrijke instellingen meegenomen.

Class Weights

In de loss functie zijn class-weights meegenomen. Class-weights zorgen ervoor dat de loss wordt vermenigvuldigd met de gegeven factor. Dit geeft de mogelijkheid om bij ongebalanceerde datasets het model extra te sturen op foutieve/onzekere classificatie voorspellingen. Voor de class weights wordt de inverse gebruikt.

Ray Tune metric

In plaats van het minimaliseren van de test_loss (standaard instelling), wordt er gefocused op het vinden van de grootste recall waarde.

Hyperparameter keuzes

Hidden size

De grootte van de hidden sizes zeggen iets over de complexiteit van het probleem. Omdat er momenteel geen domein-expert beschikbaar is wordt er gekozen een brede range te nemen tussen de 8 – 200 units. Omdat er in beide modellen gebruik wordt gemaakt van een vorm van MultiHead Attention is het noodzakelijk dat het aantal hidden layer deelbaar moet zijn door het aantal heads (PyTorch, n.d.).

Dropout

Voor de dropout hyperparameter wordt een conservatieve range van 0.1 tot 0.5 gekozen, gebaseerd op de aard van ECG-aritmie classificatie. Dropout werkt door het uitschakelen van neuronen tijdens training, waardoor het model gedwongen wordt om andere patronen te leren in plaats van te vertrouwen op specifieke neuron-combinaties. De gebruikelijke rates voor bijvoorbeeld CNN's zitten meestal tussen de 0.3 – 0.7 (Peng, 2024). Omdat wordt verwacht dat kleine variaties in de ECG-signalen moeten gaan leiden tot een correcte classificatie, wordt verwacht dat een te hoge dropout rate tot het verliezen van patronen zal leiden.

Resultaten en Analyse

Bij het evalueren van de modelprestaties wordt primair gefocust op de totale recall scores van beide architecturen, gecombineerd met een analyse van de training en test losses om mogelijke overfitting te identificeren. De recall metric is van cruciaal belang voor dit classificatieprobleem, aangezien het vermogen om alle klassen correct te identificeren van groter belang is dan enkel de algemene nauwkeurigheid. Door de train en test losses naast elkaar te analyseren, kan worden vastgesteld of de modellen daadwerkelijk generaliseren naar nieuwe data of dat er sprake is van overfitting wanneer de training loss significant lager is dan de test loss.

CNN Resultaten

De CNN Architectuur heeft gedurende 150 trials verschillende hyperparameter combinaties getest binnen een zoekruimte van hidden sizes tussen 40-150 units, 1-2 lagen, en drie verschillende activatiefuncties (GELU, ReLU, LeakyReLU). De resultaten tonen aan dat het beste presterende CNN-model een uitzonderlijk hoge recall score van 0.965 behaalde met een compacte architectuur van 64 hidden units, één laag en LeakyReLU activatie.

Trial	Hidden Size	Layers	Activation	Recall	Accuracy	Train Loss	Test Loss
train_8895675c	64	1	LeakyReLU	0.965	0.969	0.065	0.134
train_e7a9fa83	96	1	LeakyReLU	0.961	0.962	0.079	0.122
train_4bab8f4c	104	1	LeakyReLU	0.961	0.965	0.068	0.191
train_b7b133fc	104	1	GELU	0.954	0.961	0.079	0.197
train_ff3dd25a	72	1	GELU	0.952	0.966	0.07	0.198
train_f1c75817	136	1	LeakyReLU	0.949	0.963	0.058	0.226
train_8baf8aa6	112	1	LeakyReLU	0.949	0.959	0.091	0.242

Figuur 1: Overzicht van de top 7 best presterende CNN-architectuur trainingsruns, gesorteerd op hoogste recall. De tabel toont onder andere het aantal verborgen units (Hidden Size), activatiefunctie, en de bijbehorende scores op recall, accuracy, train loss en test loss. Opvallend is dat de LeakyReLU-activatie dominant is onder de best presterende modellen.

Een blik op de loss toont dat de training losses laag zijn (0.058-0.091), terwijl de test losses verhoudingsgewijs hoger liggen. Dit patroon suggereert dat de CNN-architecturen effectief leren van de training data zonder te extreme overfitting, hoewel er enige mate van overfitting aanwezig is gezien het verschil tussen train en test losses. Een kritieke observatie is dat alle top presterende modellen exclusief één convolutionele laag gebruiken, wat impliceert dat de inherente complexiteit van het hartaritmie patroonherkenningsprobleem optimaal wordt aangepakt door relatief eenvoudige maar goed geoptimaliseerde architecturen. LeakyReLU emergeert als de dominante activatiefunctie in 5 van de 7 top resultaten, wat duidt op de effectiviteit van deze functie voor het vermijden van de dying ReLU problematiek in dit specifieke domein. Bij dying ReLU's kunnen bepaalde neuronen permanent inactief gezet omdat ze geen feedback meer ontvangen tijdens de training sessies. De LeakyReLU's zorgen ervoor dat deze neuronen nog getraind kunnen worden, wat nodig is bij de lage hidden sizes.

RNN Resultaten

De RNN-architecturen omvatte zowel LSTM als GRU-varianten over 150 trials, met hidden sizes variërend van 8-200 units, 1-2 lagen, en verschillende dropout configuraties. De beste presterende RNN-modellen tonen lagere recall scores dan de CNN-varianten, met de top performer die een recall van 0.939 behaalt met een GRU-architectuur van 168 hidden units.

Trail	Model Type	Hidden Size	Layers	Dropout	Attention Dropout	Recall	Accuracy	Train Loss	Test Loss
train_3bbb4c55	GRU	168	2	0.425	0.358	0.938	0.939	0.195	0.216
train_a6089e30	GRU	144	2	0.415	0.228	0.936	0.932	0.22	0.207
train_23373a00	GRU	136	2	0.407	0.23	0.934	0.931	0.191	0.212
train_497e6270	GRU	152	2	0.466	0.243	0.931	0.925	0.218	0.253
train_0000daf9	GRU	136	2	0.413	0.149	0.93	0.921	0.212	0.232
train_289a8644	GRU	152	2	0.21	0.244	0.929	0.923	0.2	0.239
train_20b37526	GRU	160	2	0.287	0.31	0.924	0.923	0.226	0.242

Figuur 2: Overzicht van de top 7 best presterende RNN-architectuur trainingsruns, gesorteerd op hoogste recall. De tabel toont onder andere het aantal verborgen units (Hidden Size), modeltype en aantal layers samen met de bijbehorende scores op recall, accuracy, train loss en test loss.

De loss analyse van de RNN-modellen duidt op mindere malen van overfitting in vergelijking met de CNN-modellen, waarbij de verschillen tussen training losses (0.191-0.226) en test losses (0.207-0.253) aanzienlijk kleiner zijn dan bij de CNN-modellen. Dit duidt op een betere capaciteit om de onderliggende patronen in hartaritmie data te leren zonder overdadig memoriseren van training specificaties. Een opvallende bevinding is de absolute dominantie van GRU-architecturen in alle topposities, wat suggereert dat de eenvoudigere gating mechanismen van GRU effectiever zijn dan de meer complexe LSTM-structuren voor dit specifieke probleem. Alle beste modellen gebruiken consistent 2-laags architecturen met hidden sizes in het bereik van 136-168 units, wat een optimale balans lijkt te bieden tussen modelcapaciteit en voorspellend vermogen.

Conclusie en Reflectie

Conclusie

De resultaten bevestigen grotendeels de initiële hypothese dat de Convolutional Neural Networks (CNN's) effectiever zouden zijn dan Recurrent Neural Networks (RNN's) voor hartaritmie classificatie. CNN-architecturen behaalden superieure prestaties met de hoogste recall score van 0.965 en accuracy van 0.969, vergeleken met de beste RNN prestatie van 0.938 recall. Deze bevindingen valideren de theoretische verwachting dat CNN's krachtige patroonherkenningscapaciteiten kunnen realiseren in getransformeerde ECG-representaties, ondanks het verlies van expliciete temporele informatie door de transformatie naar twee-dimensionale matrices.

Echter, verdere evaluatie van de resultaten onthult een kritiek onderscheid tussen beide architecturen. CNN modellen vertonen overfitting met verschillen tussen training losses (0.058-0.091) en test losses (0.122-0.242), terwijl RNN modellen superieure generalisatie demonstreren met minimale discrepanties tussen training (0.191-0.226) en test losses (0.207-0.253).

Aan de hand van de resultaten kan een CNN-netwerk gebruikt worden met een enkele laag met een hidden size van ~64 met een LeakyReLU activation layer.

Reflectie

Een belangrijk aspect in van dit onderzoek in vergelijking met de andere onderzoeken uit deze cursus is het ombouwen van de hypertuning scripts om zich te laten focussen op Recall in plaats van test_loss. Mijn grootste zorgt in dit vlak is dat dit zou gaan leiden tot overfitting, wat het in het geval van de Convolutional Neural Networks ook heeft gedaan. In een vervolgonderzoek zou ik verder literair onderzoek doen in wat voor metrics ik zou kunnen gebruiken om in dit geval de overfitting te kunnen voorkomen, doormiddel van het maken van een custom metric.

Initieel had ik verwacht dat de accuracy lager zal uitvallen omdat er vaak is gesproken over de Accuracy / Recall tradeoff, alleen valt deze in de praktijk reuze mee.

Bibliography

Kapila, N., Glattki, J., & Rathi, T. (2024). *CNNtention: Can CNNs do better with Attention?*

Peng, C. (2024). *Comprehensive Analysis of the Impact of Learning Rate and Dropout Rate on the Performance of Convolutional Neural Networks on the CIFAR-10 Dataset*. Beijing: EWA Publishing.

PyTorch. (n.d.). *Multihead Attention*. Retrieved from docs.pytorch: https://docs.pytorch.org/docs/stable/generated/torch.nn.MultiheadAttention.html