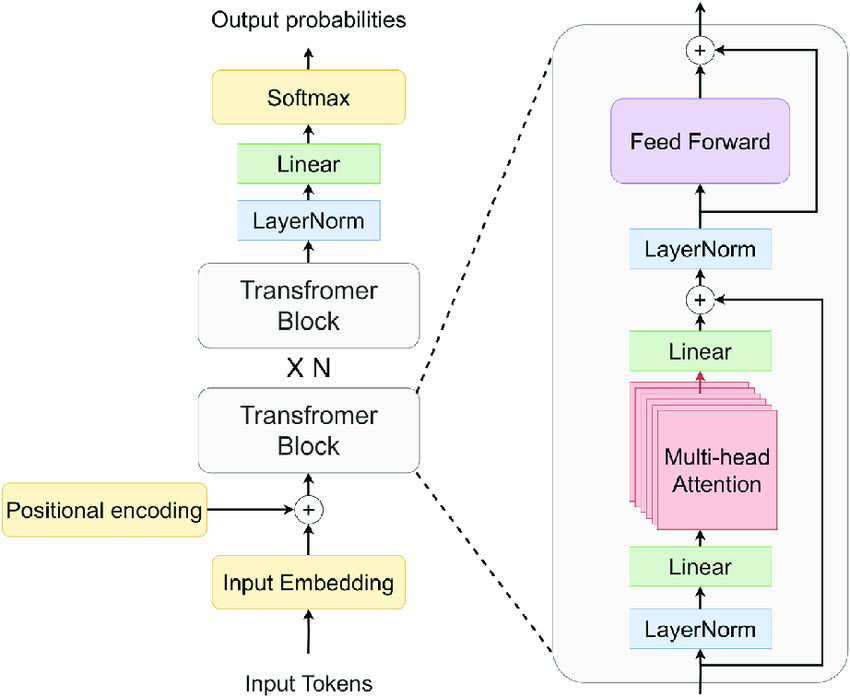
Research Question: What impact do different layer normalisation methods (Post-LN vs. Post-PN vs. Post-BLN) have on the training effectiveness and performance of decoder-only transformer architectures (e.g. GPT-2)?



| (100 words)  Introduction - Deep Learning and Decoder-only Transformers  Deep learning, a sophisticated subset of machine learning, utilizes layered neural networks to interpret vast data arrays, identifying complex patterns and enhancing decision-making with minimal human intervention. This technology powers advancements in various applications, including the development of recent chatbots like ChatGPT. Inspired by such innovations, this research delves into decoder-only transformers, a specific architecture type within deep learning.  These models, which forego traditional encoding stages, have revolutionized fields by directly processing sequential data, prompting an exploration of their unique capabilities and optimizations in this study.  Deep learning, a subset of machine learning, employs multi-layered neural networks to analyze extensive data sets, automating the construction of analytical models. As the most advanced branch of machine learning, it enables systems to independently learn from data, discern complex patterns, and facilitate decisions with minimal human input.  This technology drives advancements across various domains, including the evolution of recent chatbots like ChatGPT. Inspired by these breakthroughs, this study delves into decoder-only transformers, which are integral in natural language processing (NLP). The research aims to explore the architectural nuances and the potential for optimizing these models to better handle complex linguistic structures and improve interpretative capabilities. |
| --- |
| (400 words)  Background Information - How decoder-only transformers work   * Tokenizer and Positional embedding * Q, K, V and self attention ++ attention is all you need paper * non-linear (very brief introduction) * Feedforward (very brief) * layer normalisation, briefly touch on internal covariate * backpropagation, optimisers |
| (175 words)  Background Information - Layer Normalisation   * why LN needed * brief history on CNN + RNNs + BN (very brief only words) |
| (1000 words)  Theoretical Background - LN   * what does LN do (maths stuff begins)   + im expanding upon the work of BLN, but applying PN instead * RMSnorm vs PN vs LN vs BLN/PLN (m)   + explain their differences   + show architecture, differences   >> Formulates hypothesis here, based on differences, etc. |
|  |
| (500 words)  Experimental methodology and materials   * GPUs → Ubuntu, Nvidia A100, 96x-AMD-CPUs * PyTorch, implemented GPT-2 based on Karparthy’s course   + my modifications, using GPT-4 tokenizer instead. why? (150 words)   + torch.compile explanation ← speeding up (50 words?) ++ FLASH attention, CUDA speed up   + gradient checkpointing, HF   + Datasets → fineweb.edu     - upload to HF   + model sizes (n-embd) 128m, 753m, 1.5B   + sequence length – 1024 to 4096 * Implemented layer normalisation by code, and using sIncerass’ code, and facebook’s RMS code, and personal implementation of BLN (substituting) |
| Experimental Results and Reflection |
| (1550 words)  Analysis   * Evaluating metrics   + BLEU,   + hellaswag?   + fine-tuning for a few tasks * Raw Result Analysis * Experimental Analysis and Limitations * Conclusion |
| (275 words)  Evaluation and Improvements   * Further improvements   + larger models, more variations, larger token context   + better fine-tuning ?   + multi-modal ? (a lot more compute will be needed) * Evaluation   + better fine-tuning techniques probably will cause more impacts   + probably futile to be doing little optimisations like this? as better to instead wait for even stronger compute, allowing for newer architectures to be used |
|  |