Google

Summer of Code

Proposal - Google Summer of Code 2025

By Mayukh Tunga

Table of Contents

---

1. Synopsis

This project aims to create beginner-friendly, educational examples of large language models (LLMs) implemented in JAX and Flax. By developing clear, modular, and reproducible reference implementations of popular transformer architectures (e.g., a miniGPT-style model), the project will provide practical notebooks and training/inference scripts that other researchers and developers can build upon. The work will include detailed documentation to demystify model components and foster a deeper understanding of LLMs through hands-on examples.

---

2. Benefits to the Community

- **Educational Value:**
  The project will produce clean, well-commented Jupyter notebooks and code examples that illustrate key concepts in LLM development with JAX and Flax. This resource will be particularly valuable for developers seeking a modern alternative to PyTorch/TensorFlow examples.
- **Research and Experimentation:**
  By providing reproducible and reference implementations, researchers can quickly experiment with modifications, benchmark performance, and integrate pretrained weights from existing models.
- **Open-Source Contribution:**
  Publishing these examples on GitHub will contribute to the open-source ecosystem, supporting community collaboration and the continuous improvement of JAX and Flax for LLM work.

---

3. Deliverables

1. **Codebase:**
   - A modular Flax implementation of a GPT-style language model.
   - Integrated tools for loading and validating pretrained weights (using, for example, HuggingFace conversion tools).
   - Utility modules for autoregressive text generation, including attention mechanisms, feed-forward blocks, and positional encodings.
2. **Educational Notebooks:**
   - Jupyter notebooks that walk through each component of the model from data preprocessing and tokenization to training and inference.
   - Step-by-step explanations and visualizations to help users understand the implementation and performance optimization.
3. **Documentation and Testing:**
   - Detailed documentation covering installation, usage instructions, and contribution guidelines.
   - Unit tests for core modules to ensure reproducibility and code reliability.
4. **Benchmarks and Evaluation:**
   - Preliminary performance benchmarks on TPU/GPU environments.
   - A comparative analysis of inference time and memory utilization to guide best practices.

---

4. Timeline

Community Bonding Period (May 8 – June 1)

- Orientation: Familiarize with JAX/Flax best practices and review reference implementations (e.g., miniGPT tutorial).
- Environment Setup: Configure local, Colab, and TPU environments.

Coding Phase 1 (June 2 – July 14)

- Develop a basic GPT-style model in Flax with a clear modular structure.
- Implement data preprocessing and tokenization pipelines.
- Create initial Jupyter notebook with a walkthrough of the model, training loop, and inference.

Coding Phase 2 (July 19 – August 25)

- Enhance the model by integrating pretrained weight loaders and advanced features (e.g., multi-head attention optimizations).
- Extend the notebooks to include performance benchmarking and optimization tips.
- Finalize documentation and conduct thorough testing.

Final Submission (August 25 – September 1)

- Complete the final GitHub repository version with all examples, notebooks, documentation, and performance reports.
- Address mentor feedback and submit the finalized work product.

---

5. Technical Approach

Modular Architecture

- **Reusable Blocks:** Develop core Flax modules for multi-head self-attention, feed-forward layers, and positional embeddings to facilitate easy modifications and reuse.
- **Weight Loading and Conversion:** Create scripts to convert and load weights from published models, potentially using HuggingFace tools to ensure compatibility with Flax.
- **Inference Module:** Build a simple autoregressive decoder optimized with JAX's just-in-time compilation (jit) for efficient inference.
- **Scalability:** Utilize `jax.pmap` to support multi-device (TPU/GPU) training and inference, ensuring the approach is scalable.

Educational Notebooks and Documentation

- Develop detailed notebooks that explain each part of the model, from tokenization to the training loop.
- Include interactive visualizations and commentary to facilitate understanding of the underlying concepts and the advantages of JAX/Flax.

Expanding Existing Examples

This project will extend the miniGPT tutorial by:

- **Multi-Device Training:** Adding notebooks that demonstrate scaling the miniGPT example to **multi-GPU/TPU setups** using JAX's pmap and shard_map, with explanations of sharding strategies and performance tradeoffs.
- **Advanced Attention Variants:** Implementing and documenting optimized attention mechanisms (e.g., **FlashAttention**, memory-efficient attention) within the existing miniGPT architecture to showcase JAX's flexibility for cutting-edge research.

- **Optimization Workflows:** Expanding the tutorial with sections on **gradient checkpointing**, mixed-precision training, and JAX's remat to reduce memory usage during training.
- **Real-World Use Cases:** Building companion notebooks that adapt miniGPT for tasks like **domain-specific text generation** (e.g., code, poetry) or fine-tuning on custom datasets.

These enhancements will make the miniGPT tutorial more comprehensive while retaining its beginner-friendly ethos. All additions will be rigorously tested and include interactive visualizations to help users grasp distributed training and optimization concepts.

Tools and Libraries

- JAX and Flax for model development.
- Optax for optimization routines.
- HuggingFace Datasets and NumPy for data processing.
- GitHub Actions for continuous integration and testing.

---

6. About Me

- **Strong foundation in Deep Learning and Python**
  I've worked extensively with CNNs for both image and audio data during a research internship under Cheng Chung University (Taiwan) in collaboration with SRM University.
  - Built a full audio classification pipeline using PyTorch
  - Converted audio clips into mel spectrograms and trained a custom CNN (~75% accuracy)
  - Wrote my own dataloader and preprocessing logic for both image and audio datasets
- **Experience with low-level model design**
  I worked without wrappers like PyTorch Lightning, manually

handling model definition, training loop, evaluation, and data handling. Here's a simplified version of the audio CNN I designed:

```
class PineappleCNN(nn.Module):
    def __init__(self, num_classes=4):
        super().__init__()
        self.net = nn.Sequential(
            nn.Conv2d(1, 32, kernel_size=3, padding=1),
            nn.BatchNorm2d(32),
            nn.ReLU(),
            nn.MaxPool2d(2),

            nn.Conv2d(32, 64, kernel_size=3, padding=1),
            nn.BatchNorm2d(64),
            nn.ReLU(),
            nn.MaxPool2d(2),

            nn.Conv2d(64, 128, kernel_size=3, padding=1),
            nn.BatchNorm2d(128),
            nn.ReLU(),
            nn.MaxPool2d(2),

            nn.AdaptiveAvgPool2d((4, 4)),
            nn.Flatten(),
            nn.Linear(128*4*4, 256),
            nn.ReLU(),
            nn.Dropout(0.5),
            nn.Linear(256, num_classes)
        )

    def forward(self, x):
        return self.net(x)
```

- **Hackathon experience (3rd place) Link**

  I built an AI system that turns a single image into a comic strip:
  - Used CLIP to encode the image and an LLM to generate a story split into panels
  - Generated illustrations for each panel using Stable Diffusion
  - Built a simple site to render the final comic strip layout

- **Currently building a text-to-GIF model using JAX/Flax Link**
  - Using a 3D U-Net architecture with Factorized Attention for autoregressive video generation
  - Entirely implemented in JAX/Flax—directly relevant to this GSoC project
  - Actively exploring attention bottlenecks and optimizing video generation workflows
- **Open-source and teaching mindset**
  I enjoy documenting my code and making it clean and readable. I also love explaining technical concepts in simple terms—something I aim to bring into my LLM implementations for this project.

---

7. Commitment & Availability

I am committed to dedicating 35–40 hours per week throughout the GSoC period. I have reviewed the official timeline and, aside from my end-semester examinations from May 16 to May 26 (during which I will allocate 1–3 hours per day), I will be fully available during the coding phases. I will maintain close communication with mentors and the community via GitHub, Slack, and regular email updates. Additionally, I have set up a dedicated GitHub repository for this project, which will serve as the central location for all deliverables, notebooks, and documentation.

Repository: https://github.com/MayukhTunga/Gsoc2025-llm-jax-examples

---

8. Conclusion

If given the opportunity to work on this project under the mentorship of the JAX team this summer, I am committed to putting in my best effort to ensure meaningful progress and contributions. I am eager to

collaborate closely with the mentors and the broader community, and I would be glad to continue contributing even beyond the GSoC period. With a strong understanding of the technical stack and a genuine enthusiasm for learning and open source, I hope to become a dependable contributor to the project’s goals.

Thank you for considering my proposal. I am eager to collaborate with mentors and the community to bring this project to fruition.
