Vector Extensions

DeepSpeech heavily relies on **matrix multiplications, LSTM computations, activations, and FFT/STFT-based feature extraction**.

| **DeepSpeech Operation** | **Needed Vector Instructions** | **Covered?** |
| --- | --- | --- |
| **Matrix Multiplication (Wx + b)** | vsetvli, vld, vst, vadd, vmul | ✅ Yes |
| **LSTM Gates (Wx + Uh + b)** | vfmul, vfadd, vfsub, vfmacc | ✅ Yes |
| **Batch Normalization** | vsub, vmul, vdiv, vredsum | ✅ Yes |
| **Activation Functions (Sigmoid, Tanh)** | vfmul, vfadd, vfsub, vredsum | ✅ Yes (but needs LUT for Sigmoid/Tanh) |
| **FFT / STFT** | vfmul, vfadd, vperm (for bit-reversal) | ✅ Mostly (optimized FFT may need custom instructions) |
| **Log-Softmax for CTC Loss** | vlog, vredsum | ❌ Missing vlog (would need lookup tables) |

**🔹 Are These Vector Instructions Enough?**

✅ **For Matrix Multiplications & LSTM Computations:** YES, the RVV instructions cover everything required.  
✅ **For Activation Functions (Sigmoid, Tanh, ReLU):** **Partially covered** but may need **LUT-based implementations** since vlog is missing.  
✅ **For FFT Acceleration:** **Mostly covered**, but optimized **FFT needs specialized permutations** (vperm helps).  
❌ **For Log-Softmax (CTC Loss):** Lacks vlog, so it **needs a lookup table (LUT) implementation.**

**🔹 Recommended RVV Optimizations**

If you stick **only to standard RVV instructions**, here’s how you can optimize:

**1️⃣ Matrix Multiplications (Wx + b)**

Use **vectorized dot-product operations**:

assembly

CopyEdit

vsetvli t0, a0, e32, m8 # Set vector length

vld v0, (a1) # Load matrix row

vld v1, (a2) # Load vector

vmul v2, v0, v1 # Multiply element-wise

vredsum v3, v2 # Sum elements for dot product

✅ **Fully covered by your RVV instructions!**

**2️⃣ LSTM Gate Computation (sigmoid(Wx + Uh + b))**

LSTM needs **fused multiply-accumulate (vfmacc)** to efficiently compute:

ht=σ(Wxt+Uht−1+b)h\_t = \sigma(Wx\_t + Uh\_{t-1} + b)ht​=σ(Wxt​+Uht−1​+b)

assembly

CopyEdit

vld v0, (a1) # Load weights W

vld v1, (a2) # Load input vector x

vmul v2, v0, v1 # Wx

vld v3, (a3) # Load recurrent weights U

vld v4, (a4) # Load previous hidden state h

vfmacc v2, v3, v4 # Wx + Uh

vadd v2, v2, v5 # Add bias b

✅ **Fully covered using vfmacc!**

**3️⃣ Sigmoid & Tanh Approximation (LUT-based)**

Since sigmoid(x) = 1 / (1 + e^-x), RVV lacks vlog and vexp.  
🔹 **Solution:** Use **LUT-based approximation for sigmoid & tanh**.

assembly

CopyEdit

vld v0, (a1) # Load precomputed sigmoid LUT

vperm v1, v0, v2 # Approximate sigmoid(x) using LUT

✅ **Covered, but requires a precomputed LUT.**

**4️⃣ FFT / STFT Acceleration**

FFT requires:  
🔹 **Complex Multiplication** → Covered using vfmul, vfadd, and vfsub.  
🔹 **Bit-Reversal Permutations** → **vperm helps** but may need an extra shuffle instruction.

✅ **Mostly covered, but highly optimized FFT might need custom instructions.**

**5️⃣ Log-Softmax for CTC Loss**

Since Log-Softmax requires log(x), and **RVV does not provide vlog**, we must use:

* **LUT-based log(x) approximation.**
* **vredsum for computing sum of exponentials**.

❌ **Partially covered, requires a LUT-based workaround.**

**🔹 Final Verdict:**

| **DeepSpeech Component** | **RVV Support** | **Additional Optimizations Needed?** |
| --- | --- | --- |
| **Matrix Multiplications** | ✅ Fully covered | - |
| **LSTM Gate Computation** | ✅ Fully covered | - |
| **Batch Normalization** | ✅ Fully covered | - |
| **Activation Functions (Sigmoid, Tanh)** | ⚠️ Partially covered | LUT-based Sigmoid/Tanh |
| **FFT / STFT** | ⚠️ Mostly covered | Bit-reversal optimization needed |
| **Log-Softmax (CTC Loss)** | ❌ Not covered | Requires LUT-based log(x) |

🚀 **Yes, RVV is mostly enough!** But **you need LUT-based implementations for Sigmoid, Tanh, and Log-Softmax.**