ARQUITECTURA MusicLSTM

Datos de Entrada

Input 1: Pitch_seq

Tipo: LongTensor
Shape: (Batch, 32)
Rango: [0, 127] (notas MIDI)
Naturaleza: Variable categórica

Input 2: Cont_seq

Tipo: FloatTensor
Shape: (Batch, 32, 3)
Columnas:
- [0] = step (normalizado)
- [1] = duration (normalizado)
- [2] = velocity (normalizado)

Capa 1: Embedding

Convierte índices discretos de pitch (0-127) en vectores densos de 96 dimensiones.
Ventaja: El modelo aprende relaciones semánticas entre notas (ej: Do# cerca de Do).

nn.embedding

- **vocab_size:** 128
- **embedding_dim:** 96
- **Parámetros:** 128 × 96 = 12,288

- Dropout(0.15)
- Regularización aplicada post-embedding para prevenir memorización de secuencias específicas.

Output: (B, 32, 96)

Concatenación Multi-Modal

Decisión de diseño: Se fusionan ANTES de la LSTM para que la red aprenda interacciones entre pitch y propiedades temporales desde el inicio

```
x = torch.cat([emb_output, cont_seq], dim=-1)
```

Fusiona:
• Embeddings de pitch (96D)
• Features continuas (3D)
= Vector unificado de 99D por timestep

Resultado: Cada uno de los 32 timesteps ahora tiene 99 features (96 de embedding + 3 continuas)

Output: (B, 32, 99)

Modelo LSTM (3 capas apiladas)

Elección de LSTM
• Captura dependencias de largo plazo (memoria)
• Previene gradiente desvaneciente
• Estado del arte para secuencias musicales

nn.embedding

```
nn.LSTM(
  input_size=99,
  hidden_size=320,
  num_layers=3,
  dropout=0.35,
  batch_first=True
)
```

Capa 1	Capa 2	Capa 3
(99 → 320) = 171,520 params	(320 → 320) = 410,880 params	(320 → 320) = 410,880 params

Output: (B, 32, 320)

Dropout 0.35: Aplicado entre capas para regularización

Extracción Último Timestep

Output: (B, 320)

last_hidden = lstm_output[:, -1, :]

Toma solo el último hidden state que contiene
toda la información contextual de las 32 notas
Resultado: Pasamos de una secuencia (32 timesteps) a un único vector de contexto (320 dimensiones)

Layer Normalization

nn.LayerNorm(320)

$x\_norm = (x - \mu) / \sqrt{(\sigma^2 + \varepsilon)}$
$output = \gamma \odot x\_norm + \beta$

Parámetros aprendibles: $\gamma, \beta \in \mathbb{R}$

Beneficios:
• Estabiliza el entrenamiento
• Reduce internal covariate shift
• Permite learning rates más altos
• Acelera convergencia

Output: (B, 320)

División en Dos Cabezas

Cabeza 1: Pitch (Clasificación)

nn.Linear(320 → 128)

Parámetros:
W: (128, 320) = 40,960
b: (128,) = 128
Total: 41,088

Output: (B, 128) logits.
Sin activación (raw logits para CrossEntropy)

Output: (B, 128)

Cabeza 2: Continuos (Regresión)

nn.Linear(320 → 3)

Parámetros:
W: (3, 320) = 960
b: (3,) = 3
Total: 963

Output: (B, 3) valores reales normalizados.
Sin activación (regresión)

Output: (B, 3)

Salidas del Modelo

SALIDA 1: pitch_logits

Tipo: FloatTensor
Shape: (B, 128)

Interpretación: Logits sin normalizar para cada nota MIDI (0-127)

SALIDA 2: cont_pred

Tipo: FloatTensor
Shape: (B, 3)

Interpretación:
[0] = step (norm)
[1] = duration (norm)
[2] = velocity (norm)