Documento técnico: `Multi-horizon LSTM` — descripción, flujo de datos y detección de gradientes

Generado a partir del texto del proyecto

1 de noviembre de 2025

Resumen

Resumen del sistema de entrenamiento multi-horizon, su pipeline de prueba, la detección de gradientes y el diagrama de flujo de datos. Incluye las fórmulas de pérdida principales y pseudocódigo del loop de entrenamiento, así como un checklist de tareas y ejemplos de logs.

1. Mejoras de rendimiento esperadas

Basado en el análisis *quant-research-architect*, se estiman las siguientes mejoras:

- **+20–40 %** de *directional accuracy* en horizontes no nativos (vs. scaling).
- **+20–30 %** en *confidence interval coverage accuracy*.
- **+10–20 %** en precisión a corto plazo (horizontes $h = 1, 3$) usando *inverse variance weighting*.

2. Cómo probar (How to Test)

1. Iniciar la aplicación `TradeApp`:

```
py -3.10 TradeApp.py
```

2. En la GUI:
   - Ir a la pestaña `Training`.
   - Expandir "Training Config".
   - Activar **Enable Multi-Horizon**.
   - Configurar opciones o usar valores por defecto.
   - Pulsar `Prepare + Train (background)`.

3. Monitorizar:
    - Buscar en logs: "Computing inverse variance weights..." en la época 2.
    - Ver el breakdown por horizonte cada 5 épocas.

3. Checklist (Todos)

☒ Implementar sistema de pérdida modular en `fiboevo.py` (`mse_loss_component`, `heteroscedastic_nll_component`, `multihorizon_loss_configurable`).

☒ Actualizar firma `train_multihorizon_lstm()`.

☒ Implementar weighting adaptativo (cálculo de inverse variance tras época 1).

☒ Implementar detección de *head interference*.

☒ Añadir logging per-horizon.

☒ Validar sintaxis de `fiboevo.py`.

☒ Añadir panel de configuración multi-horizon en GUI.

☒ Implementar `_train_model_multihorizon()` con parseo completo.

☒ Validar sintaxis de `TradeApp.py`.

☐ Test multi-horizon vía GUI con distintas configuraciones de pérdida.

☐ Mejorar status dashboard con selector de horizon y métricas per-horizon (Fase 3).

4. Detección de gradientes (Gradient Monitoring)

4.1. Tipos de detección

1. **Detección de gradientes explosivos**. Se calcula la norma total de gradientes:
$$\text{total\_norm} = \sqrt{\sum_{\theta} \|\nabla_\theta L\|^2}$$
y se aplica `clip_grad_norm_` con un umbral `grad_clip`. Además, si $\text{total\_norm} > 3 \times \text{grad\_clip}$ se emite un warning.

2. **Detección de interferencia entre heads**. En cada época (si habilitado) se compara la pérdida por horizon con la época previa; si
$$\text{loss}_h^{(t)} > (1 + \text{threshold}) \cdot \text{loss}_h^{(t-1)}$$
se emite un warning de *head interference*.

4.2. Fragmento de código (esquema)

```
if grad_clip:
    total_norm = nn.utils.clip_grad_norm_(model.parameters(),
        grad_clip)
    if total_norm > grad_clip * 3:
        LOGGER.warning(f"Large gradient detected: {total_norm:.2f} (
            clip={grad_clip})")
```

5. Diagrama de flujo de datos (arquitectura y forward/backward)

5.1. Arquitectura (esquema)

```
INPUT SEQUENCE
  [B, seq_len=128, F=14]
       |
  SHARED LSTM ENCODER (2 layers, 128 hidden)
       |
  LAST HIDDEN [B, 128]
    / | \ ... \
 HEAD1 HEAD3 ... HEAD24
 (MLP return) (MLP return) (MLP return)
 (MLP vol + softplus) ...
       \ | /
   PREDICTIONS DICT {1:(ret1,vol1), 3:..., 24:...}
```

5.2. Pérdida multihorizon configurable

Para cada horizon h:

- **Heteroscedastic NLL** (si se elige):

$$\mathrm{var}_h = \sigma_h^2, \quad \mathrm{NLL}_h = \tfrac{1}{2}\big(\log \mathrm{var}_h + \tfrac{(y_h - \hat{y}_h)^2}{\mathrm{var}_h}\big)$$

  posibilidad de penalización direccional:

$$\mathrm{sign\_penalty} = \frac{1 - \mathrm{sign}(\hat{y}_h)\,\mathrm{sign}(y_h)}{2}$$

  y $\mathrm{loss}_h = \mathrm{NLL}_h + \alpha_{\mathrm{dir}} \cdot \mathrm{sign\_penalty}$ (si aplica).

- **MSE**:

$$\mathrm{loss}_h = \mathrm{MSE}(y_h, \hat{y}_h) + \alpha_{\mathrm{vol}}\,\mathrm{MSE}(\sigma_h, \hat{\sigma}_h)$$

- **Híbrida**: $\mathrm{loss}_h = 0{,}5 \cdot \mathrm{MSE} + 0{,}5 \cdot \mathrm{NLL}$.

5.3. Agregación ponderada (ejemplo de inverse variance weighting)

Pesos (ejemplo):

$$\text{weights} = \{1:2{,}5,\ 3:1{,}8,\ 6:1{,}0,\ 12:0{,}6,\ 24:0{,}4\}$$

y la pérdida total:

$$\mathcal{L} = \sum_h w_h \cdot \text{loss}_h.$$

5.4. Backward / clipping

- Backprop: $\nabla_\theta \mathcal{L}$.
- Suma de gradientes hacia el encoder: los gradientes de cada head se suman en los parámetros del encoder.
- Clip por norma: si total_norm $>$ thresh se escala todo.

6. Detección de interferencia — ejemplo

Se comprueba por horizon si la pérdida aumentó $>10\,\%$ entre épocas consecutivas. Si es así, generar warning y registrar detalles.

7. Pseudocódigo resumido del loop de entrenamiento

```
for epoch in range(epochs):
    if use_adaptive_weights and epoch == 1:
        compute_inverse_variance_weights()
    for batch in train_loader:
        predictions = model(X_batch)
        loss, losses_h = multihorizon_loss_configurable(predictions,
            targets, computed_weights)
        loss.backward()
        total_norm = clip_grad_norm_(model.parameters(), 1.0)
        if total_norm > 3.0:
            WARN
        optimizer.step()
    val_loss, val_losses_h = eval_epoch()
    if detect_interference and epoch > 5:
        check_per_horizon_increases()
```

8. Ejemplo de logging por epoch

```
Epoch 5/50 - Per-Horizon Breakdown
Horizon  Train Loss    Val Loss      %
h=1      0.001200      0.001250      4.2%
...
TOTAL    0.004220      0.004450
Best Val: 0.004220 | Patience: 0/10
WARNING: HEAD INTERFERENCE: Horizon 6 loss increased 15.2%
WARNING: Large gradient detected: 4.25 (clip=1.0)
```

Fin del documento LaTeX