**Advanced AI Forecasting with TensorFlow and Natural Language Processing**

**Week 4 Instructor Lecture Notes – Exploring different Keras Layers for different lookback periods**

**Reminder:** A complete, cell‑by‑cell walkthrough of the shared notebook pipeline (data loading, feature engineering, scaling, windowing, splits, training loop, evaluation, saving) is already covered in the Week 3 Instructor Lecture Notes. For Week 4, all six notebooks are virtually identical in those cells; the only material differences are the lookback window and the model configuration described below. Use Week 3 notes whenever you need to explain non‑model cells**.**

**Quick orientation for students**

* This week materials explore how sequence length (lookback) and model depth interact.
* Longer lookbacks = more historical context but higher risk of diluting signal and overfitting
* Shorter lookbacks = more reactive to recent data but risk missing slower trends.
* Capacity (Conv1D + stacked BiGRUs + Attention) is right‑sized to each lookback to balance bias, including variance, training time, and generalization.

**Model differences at a glance**

**All models share the same input shape (lookback, features) and the same tail: SimpleAttention(128) → Dense(1).**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lookback | Front‑end | Temporal stack | Attention | Head |
| 270 days | Conv1D(64, k=3, relu) | 3× Bidirectional(GRU(250, return\_sequences=True, dropout=0.2) | SimpleAttention(128) | Dense(1) |
| 180 days | Conv1D(64, k=3, relu) | 4× Bidirectional(GRU(250, return\_sequences=True, dropout=0.2) | SimpleAttention(128) | Dense(1) |
| 90 days | Conv1D(64, k=3, relu) | 2× Bidirectional(GRU(250, return\_sequences=True, dropout=0.2) | SimpleAttention(128) | Dense(1) |
| 60 days | Conv1D(64, k=3, relu) | 2× Bidirectional(GRU(250, return\_sequences=True, dropout=0.2) | SimpleAttention(128) | Dense(1) |
| 30 days | Conv1D(64, k=3, relu) | 2× Bidirectional(GRU(250, return\_sequences=True, dropout=0.2) | SimpleAttention(128) | Dense(1) |
| 14 days | Conv1D(64, k=3, relu) | 2× Bidirectional(GRU(250, return\_sequences=True, dropout=0.2) | SimpleAttention(128) | Dense(1) |

**Shared inductive biases**

* **Conv1D(64, k=3):** catches local motifs (weekly/monthly bursts, volatility clusters) before recurrent modeling.
* **BiGRU(250) stacks:** model temporal dependencies; bidirectionality uses the full window context during training/inference.
  + Last BIGRU layer uses kernel\_regularizer=regularizers.l2(1e-5).
    - This allows you to add weight regularization to layers (e.g., Dense, Conv1D, GRU) to help reduce overfitting.
* **SimpleAttention(128):** learns which days in the window matter most, instead of averaging.

**Why these layers for each lookback**

**Below are talking points you can present verbatim while showing model.summary() in each notebook.**

**270‑day lookback — Long horizon, moderate depth**

**Architecture: Conv1D(64) → 3× BiGRU(250) → Attention(128) → Dense(1)**

**Rationale**

* Window length (T=270) is long enough to contain quarterly/annual cycles. We keep three recurrent layers (not four) to limit overfitting on very long sequences while still offering hierarchical temporal abstraction.
* Conv1D front‑end efficiently extracts short-term motifs so the GRUs focus on longer-range structure.
* Attention prevents the model from smearing importance uniformly across 270 days; it actively emphasizes salient spans (e.g., recent regime changes).

**Overfitting considerations**

* Risk comes from both sequence length (many steps) and parameter count. We mitigate with:
  + dropout=0.2 in each GRU.
  + Consider L2 weight decay (e.g., kernel\_regularizer=l2(1e-5)) on the last BiGRU.
    - L2 weight decay is a regularization technique that penalizes large weight values by adding the sum of the squared weights to the loss function.
    - In Keras, this is applied via kernel\_regularizer=l2(...).
    - By using L2 weight decay on the last BiGRU layer, the model is encouraged to keep its weights smaller and more stable, which reduces overfitting and improves generalization.
    - In this context, adding l2(1e-5) would lightly constrain the BiGRU’s parameters, helping it focus on essential temporal patterns without relying too heavily on specific weight magnitudes.
  + EarlyStopping on val\_loss and ReduceLROnPlateau (already in the shared training cell).
  + Optionally reduce hidden size from 250 → 192 if validation diverges early.

**What to watch in curves**

* Training faster than validation after ~5–10 epochs suggests capacity > data. Expect earlier LR drops via ReduceLROnPlateau compared with shorter windows.

**180-day lookback — Medium‑long horizon, extra temporal depth**

**Architecture: Conv1D(64) → 3× BiGRU(250) → Attention(128) → Dense(1)**

**Rationale**

* With T=180, we still have seasonal motifs but less raw time context than 270. We add a fourth BiGRU to recover hierarchical richness from a shorter canvas, improving the model’s ability to compose weekly → monthly → quarterly patterns.
* The deeper stack can learn interactions (e.g., volatility × trend) that may not surface with only two layers.

**Overfitting considerations**

* Depth adds parameters, guard with:
  + Keep dropout=0.2; if needed, and can add recurrent\_dropout=0.1–0.2 to the top GRU(s).
  + Consider L2 weight decay (e.g., kernel\_regularizer=l2(1e-5)) on the last BiGRU.
    - L2 weight decay is a regularization technique that penalizes large weight values by adding the sum of the squared weights to the loss function.
    - In Keras, this is applied via kernel\_regularizer=l2(...).
    - By using L2 weight decay on the last BiGRU layer, the model is encouraged to keep its weights smaller and more stable, which reduces overfitting and improves generalization.
    - In this context, adding l2(1e-5) would lightly constrain the BiGRU’s parameters, helping it focus on essential temporal patterns without relying too heavily on specific weight magnitudes.
  + Tighten EarlyStopping patience (e.g., from 8 → 6) if you see sharp train–val divergence.

**What to watch in curves**

* Slightly slower per‑epoch than the 3‑layer model but sometimes better validation minima. If validation bounces, the model may be too deep. Try removing the lowest GRU or shrinking units.

**90‑day lookback — Quarterly focus, balanced capacity**

**Architecture: Conv1D(64) → 2× BiGRU(250) → Attention(128) → Dense(1)**

**Rationale**

* T=90 roughly spans a quarter.
* Strong balance between responsiveness and trend capture. Two BiGRUs usually suffice to model short/medium dependencies without overfitting.
* Conv1D continues providing local feature extraction (e.g., weekly turns) that GRUs can integrate.

**Overfitting considerations**

* Lower risk than longer windows. If underfitting (both train/val loss high), you can:
  + Add a third BiGRU or increase units (e.g., 250 → 320) but monitor validation closely.
  + Increase attention units to 192 for sharper focus if attention maps look too flat.
  + dropout=0.2 in each GRU.
  + Consider L2 weight decay (e.g., kernel\_regularizer=l2(1e-5)) on the last BiGRU.
    - L2 weight decay is a regularization technique that penalizes large weight values by adding the sum of the squared weights to the loss function.
    - In Keras, this is applied via kernel\_regularizer=l2(...).
    - By using L2 weight decay on the last BiGRU layer, the model is encouraged to keep its weights smaller and more stable, which reduces overfitting and improves generalization.
    - In this context, adding l2(1e-5) would lightly constrain the BiGRU’s parameters, helping it focus on essential temporal patterns without relying too heavily on specific weight magnitudes.
  + EarlyStopping on val\_loss and ReduceLROnPlateau (already in the shared training cell).

**What to watch in curves**

* Healthy models show steady improvements for 10–30 epochs with moderate LR drops.

**60‑day lookback — Bi‑monthly scope, still two layers**

**Architecture: Conv1D(64) → 2× BiGRU(250) → Attention(128) → Dense(1)**

**Rationale**

* T=60 emphasizes recent dynamics while keeping enough context for swing‑trading rhythms. Two BiGRUs keep variance in check while allowing non‑linear interactions.

**Overfitting considerations**

* If the model plateaus early, consider slightly larger batch size to stabilize gradients or a higher initial LR (e.g., 3e-3) paired with ReduceLROnPlateau.

**What to watch in curves**

* Validation should track training closely.
* Big gaps indicate too much capacity or data noise dominance.

**30‑day lookback — One‑month micro‑structure**

**Architecture: Conv1D(64) → 2× BiGRU(250) → Attention(128) → Dense(1)**

**Rationale**

* T=30 focuses on recent micro‑structure, ideal when markets are regime‑volatile and long history is less predictive. Two BiGRUs are retained for feature interactions the convolution alone won’t capture.

**Overfitting considerations**

* With small T, the sequence dimension contributes fewer effective degrees of freedom, so overfitting often comes from units rather than time. If needed:
  + Drop units to 192 or 160.
  + Increase dropout to 0.3 on the last BiGRU.

**What to watch in curves**

* Very fast convergence. If improvement stalls after ~5 epochs, try more aggressive LR schedule or reduce depth.

**14‑day lookback — Two‑week reactivity**

**Architecture: Conv1D(64) → 2× BiGRU(250) → Attention(128) → Dense(1)**

**Rationale**

* T=14 is highly reactive.
* Two BiGRUs layers retain non‑linear capacity but rely heavily on Conv1D and Attention to avoid simply memorizing the last few days.
* Attention acts as a soft gating mechanism, often concentrating on very recent steps without discarding older ones.

**Overfitting considerations**

* Highest risk of chasing noise. Suggested defenses:
  + Add recurrent\_dropout=0.2 to the top GRU.
  + Consider GaussianNoise(σ=0.01–0.05) after Conv1D during experimentation.
  + Tighten EarlyStopping; cap epochs (e.g., ≤100).

**What to watch in curves**

* Expect early val\_loss minima. If validation degrades after a small peak, reduce units or remove one BiGRU.

**Speaking notes: Why Conv1D + BiGRU + Attention works**

* **Conv1D(k=3):** fast, translation‑equivariant filters that detect local changes (breakouts, volatility clusters).
  + It’s a noise‑resistant preprocessor for RNNs.
* **BiGRU stacks:** capture temporal hierarchies
  + Lower layers track short rhythms
  + Upper layers integrate them into longer narratives.
* **Attention:** converts a sequence into a weighted summary, letting the model ignore dull segments and elevate critical intervals.

**Rule‑of‑thumb to share:** As lookback grows, you can shallow the recurrent stack (to fight overfitting) and lean on attention to pick key spans. As lookback shrinks, you often keep 2 layers but may downsize units and use stronger regularization.

**Practical diagnostics & remedies (useful during live runs)**

**Signs of underfitting (both train & val losses high):**

* Add a BiGRU layer (except 270D) or increase units to 320.
* Increase attention units to 192.
* Loosen EarlyStopping patience to allow more epochs.

**Signs of overfitting (train ↓, val ↑):**

* Reduce units (250 → 192), increase dropout to 0.3.
* Add recurrent\_dropout 0.1–0.2 on upper GRU(s).
* Introduce mild L2 (1e‑5 to 5e‑5) on GRU kernels.
* Use smaller lookback if regime is highly non‑stationary.

**Optimizer/lr tips**

* Start Adam(lr=1e‑3). If training unstable, try 5e‑4.
* Keep ReduceLROnPlateau (factor 0.5, patience 3).
* For 14/30D, consider patience 2.

**Batch size**

* 32 is a sane default. Increase to 64 for shorter lookbacks to stabilize gradients; keep 32 for longer lookbacks to avoid OOM.

**Expected behavior by lookback (set student expectations)**

* **14–30D**: quick convergence; attention weights skew toward most recent ~5–10 steps. Gains from extra depth are limited; focus on regularization.
* **60–90D:** often the sweet spot — enough context to catch swings; manageable capacity; smoother validation curves.
* **180–270D:** better at capturing macro/seasonal effects but sensitive to overfitting and concept drift; rely on attention and early stopping, and watch for validation degradation after initial gains**.**

**What "lookback days" actually means**

If you use a lookback of 365 days, each input sequence to your model covers one year of historical data before the prediction point.  
If you use a lookback of 14 days, each sequence covers only the past two weeks.

In practical terms, the X\_train.shape[1] (timesteps) changes depending on the lookback period:

* Larger lookback → more timesteps per sample
* Smaller lookback → fewer timesteps per sample

**How different lookback periods affect performance**

**1. Long lookbacks (e.g., 365, 270, 180 days)**

* **Pros:**
  + Capture long-term seasonal trends (e.g., yearly cycles, slow changes in behavior).
  + Give the GRUs more context for detecting repeating patterns across months.
* **Cons:**
  + Can introduce noise if older data is less relevant to the near future.
  + Model may become harder to train — more timesteps = more parameters to learn relationships over a longer span.
  + May overfit if the dataset is not large enough.

**2. Medium lookbacks (e.g., 90, 60, 30 days)**

* **Pros:**
  + Good balance between short-term and long-term patterns.
  + Often works well when there are monthly or quarterly cycles.
  + Less computationally heavy than very long lookbacks.
* **Cons:**
  + Might miss very long-term seasonal dependencies (like annual cycles).

**3. Short lookbacks (e.g., 14, 1 day)**

* **Pros:**
  + Very fast training, fewer parameters.
  + Good when the target is driven by short-term momentum or recent events.
  + Less risk of overfitting to long-term noise.
* **Cons:**
  + Lose seasonal or cyclical context.
  + Predictions may be myopic — reacting only to immediate fluctuations.

**In the model specifically**

* The Conv1D + Bidirectional GRUs excel at extracting both local patterns (via convolution) and temporal dependencies (via GRU memory).
* Long lookback → The GRUs have more temporal structure to learn from, but may need more regularization (e.g., dropout, early stopping).
* Short lookback → GRUs will focus on immediate past patterns, making attention less effective for long-term signals.

**Why use multiple lookbacks in practice**

Many time series problems benefit from ensembling models with different lookback periods because:

* Short lookbacks capture recent trends.
* Long lookbacks capture seasonal and macro patterns.
* The combination often gives better predictive power than any single lookback.

**Suggested live demos**

1. Attention maps: After training, pull attention\_weights and overlay on the price series within a sample window. Show how focus shifts with lookback length.
2. Depth ablation: On 90D, compare 1 vs 2 vs 3 BiGRUs — watch val\_loss and training time.
3. Regularization sweep: On 14D, toggle recurrent\_dropout and show its effect on early overfitting.
4. Lookback swap: Train 60D and 270D with identical seeds; compare convergence speed, best val\_loss, and param counts.

*(Prepare helper cells to extract attention weights and to print* model.count\_params() *for the opening slide.)*

**Talking points & FAQ**

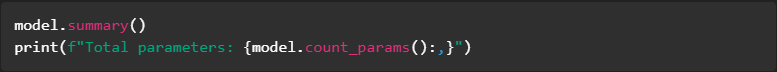
* Why bidirectional? Within the fixed window, bidirectional reads left→right & right→left, leveraging full context for representation before prediction.
* Why not Transformer? Heavier and data‑hungry; for these horizons and dataset size, GRU + attention strikes a better accuracy/compute balance.
* Do longer windows always help? No. They can dilute recent signal and invite overfitting. Use validation to choose the sweet spot.
* What if attention looks uniform? Increase SimpleAttention units to 192 or add another BiGRU layer (except 270D), then reassess.
* Production note: In real deployments, fit scalers on train only and lock them for val/test to avoid leakage.

**Appendix A – Parameter & memory intuition (no math required)**

* **Params grow with: number of layers × (hidden units)^2. Doubling units roughly quadruples RNN parameters.**
* **Memory grows with: batch × lookback × hidden × layers. Long lookbacks and deep stacks are the main levers.**
* **Training time: roughly proportional to memory footprint; 180D with 4 layers can approach 270D with 3 layers.**

**Appendix B – Optional code snippets you can drop into any Week 4 notebook**

**Print parameter count & per‑layer summary**

****

**Extract attention weights for a single batch**

**A computer screen shot of text

AI-generated content may be incorrect.**

**Try a mild L2 on the last BiGRU (experimental)**

**A screen shot of a computer program

AI-generated content may be incorrect.**

**Final reminder to instructors**

Use the Week 3 notes for every non‑model cell. In Week 4, your focus is to explain and compare how each architecture scales with its lookback window, why that scaling is sensible, and how to detect and correct under/overfitting in real time. Encourage students to justify architectural choices with validation curves and attention visualizations rather than by parameter count alone.