1D CNN of time Series data

In 1D CNN slides a small window called a Kernel across a Sequence looks for Patterns.

This 1D CNN tries to detect patterns like small rises, small drops, sudden spikes, double-top/double bottom, Local trend direction.

~~He~~ we use the Kernels & filters.

1D, 2D, 3D CNN.

1D Conv has 1 direction (time axis) to calculate convolutions.

- when a 1D Convolutional NN processes a time-Series such as past stock prices, it does not simply look entire Series as a whole. instead it breaks the Series into small windows and passes these windows through a sequence of operations — convolution, activation, pooling, flatten, and dense layers. this pipeline transform's raw price Values into rich abstract representations that the model can use to understand trends.

The Journey of the data begins at the convolution step where small learnable filters slide over the stock price sequence. If we consider a price seq [100.5, 101.2, 102.1, 101.8, 100.9, ...] Each filter moves across this sequence a few points at a time. A filter of size 3 would look at three consecutive closing price at once, calculating a weighted combination of them using its internal weights and bias. This simple operation is powerful because it allow a filter to specialize in detecting specific shapes. Each filter learns different pattern because during training, backpropagation adjusts its weights to capture whatever pattern helps reduce prediction error over time filters organize themselves into a intelligent set of pattern detector, one may 1 & f

After this step the time series is no longer just a list of numbers but has become a set of features maps, each representing how strongly a particular pattern appears at different parts of the price history.

Immediately after convolution comes the activation stage, which injects non-linearity into the model. without activation functions the CNN would behave like simple linear model incapable of capturing complex relationships in the stock movement. the most common act ReLU, in the context of stock data this has a useful interpretation. a filter might compute something slighty negative when the pattern isn't present and strongly +ve when it detects a similar shape to the one its

its learned. Relu enforces this behaviour by letting Strong signal pass forward while suppressing the weak (or) irrelevant ones. this separation of signal from noise is crucial because Kd Stock prices fluctuate constantly and the model must learn to focus only on meaningful patterns rather than being distracted

once activated, the feature maps are still too large and contain reduntant information this is where pooling plays its role: pooling compress the feature. map by taking either the maximum value in every small region a the average value. Max Pooling is particulary important for stock prediction because it preserve the strongest patterns detected in each region, ignoring fine-scale fluctuations that do not matter. ~~for~~ e pooling reduces the length of the seq makes learning faster, controls overfitting, helps CNN focus on larger, more stable structures in the data rather than noise.

After pooling the feature maps are Still two-dimensional structures with time along one axis and filters along the other side. But dense Layers that makes final predictions require a one-dimensional vector. this transformation is done by the flatten operation. flatteing takes each ~~[illegible]~~ feature map from every filter and lays them out end to end into single long feature vector with multiple filters.

280

with multiple filters each have gone convolution and pooling this flatten vector represents a very rich description of the input seq. every number in this vector represents something how strongly this filter detected this pattern in this region

Input seq (30 steps)

↓

1st Step convolution (1D Kernel)

=> Multiple filters slide across the time dimension
=> Each filter learns a pattern
=> output is feature maps.

↓

2. Activation (ReLU)

=> Negative values removed (noise reduction)
=> Strong patterns highlighted

↓

3. Pooling (Max Pool)

=> Reduce the feature map length
=> Keeps strongest pattern response
=> Loses exact time position.

↓

4. Flatten.

=> converts all pooled feature maps to one long vector
=> this vector represents learned patterns.

↓

5. Dense Layers.

=> Combine all detected patterns
=> Learn relationships between patterns and future values and produce the final prediction.

6 Loss Calculation.

=> Compare prediction vs actual
=> Compute Error.

↓

7. Backpropagation
=> Error flow backward
=> Dense layer weights are updated
=> Gradients flow to CNN filters.
=> filter kernels adjusts to learn better patterns.

↓

8. updated CNN filters.
=> Strong patterns get stronger
=> weak patterns get removed
=> filters become specialized

Important points

1. Convolution:
1. filters slide across time.
2. Each filter has a kernel
each filter produces one feature map

Dense Layer:
=> interprets patterns Learned by CNN
=> can have the multiple Layers.
=> final Layers have
Regression → 1
Classification → Softmax