

# Ink & Accent Color Scheme

Clean, Professional, No Backgrounds




Template Demonstration

September 20, 2025




All backgrounds are pure white - colors used only for text and accents

# Color Palette

## Primary Colors

-  Pure Black - Headers & Main Text
-  Warm Coral - Primary Accents
-  Slate Gray - Secondary Text

## Supporting Colors

-  Success Green - Positive
-  Warning Amber - Attention
-  Info Blue - Information

Note: No colored backgrounds - maintaining clean white space throughout

# Typography Hierarchy

## Large Header

### Highlighted Subheader

Regular body text appears in Pure Black

Secondary information in Slate Gray

Information callouts in Info Blue

Success messages in green

- **Primary** bullet points
  - Regular list items
  - Secondary information
- 1 Numbered items
  - 2 With different accents
  - 3 For emphasis

# Mathematical Formulas

## Attention Mechanism

$$\text{score}(h_t, \bar{h}_s) = h_t^T W_a \bar{h}_s$$

$$\alpha_{ts} = \frac{\exp(\text{score}(h_t, \bar{h}_s))}{\sum_{s'} \exp(\text{score}(h_t, \bar{h}_{s'}))}$$

$$c_t = \sum_s \alpha_{ts} \cdot \bar{h}_s$$

## Key Components

- $h_t$  - decoder hidden state
- $\bar{h}_s$  - encoder hidden states
- $\alpha_{ts}$  - **attention weights**
- $c_t$  - context vector

Clean presentation with accent colors only on key elements

# Content Blocks - No Backgrounds

## Definition

A **sequence-to-sequence** model transforms variable-length input to variable-length output.

## Example

Translation: "Hello" → "Bonjour"

## Important Note

Attention mechanisms solve the **bottleneck problem** in vanilla seq2seq models.

## Information

Modern transformers use **multi-head attention** for parallel processing.

# Code Highlighting

## Python Implementation

```
def attention(query, keys, values):  
    # Calculate scores  
    scores = torch.matmul(  
        query, keys.transpose(-2, -1)  
    )  
    # Apply softmax  
    weights = F.softmax(scores, dim=-1)  
    # Weighted sum  
    output = torch.matmul(weights, values)  
    return output, weights
```

## Key Points

- **Query** - what we're looking for
- **Keys** - what we compare against
- **Values** - what we aggregate
- **Weights** - attention distribution

Clean code presentation without distracting backgrounds

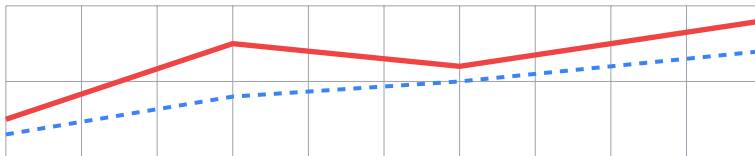
# Data Visualization Style

## Chart Guidelines

- **Primary data** in Warm Coral
- **Comparison data** in Info Blue
- Grid lines in light gray
- White background always
- Minimal borders (thin gray)

## Visual Hierarchy

- 1 **Bold titles** in Deep Ink Blue
- 2 **Key metrics** highlighted
- 3 **Annotations** in gray
- 4 **Improvements** in green
- 5 **Warnings** in amber



Sample visualization with clean style

# Comparison Layout

## Traditional Approach

- Fixed context
- Information bottleneck
- Limited range

**Performance:** 78.5%

## With Attention

- Dynamic context
- Direct connections
- Full sequence access

**Performance:** 92.3%

## Transformer

- Parallel processing
- Multi-head attention
- Positional encoding

**Performance:** 95.7%

Clean three-column comparison without background distractions



# Design Principles

## Core Principles

- 1 **No colored backgrounds** - pure white only
- 2 **Strategic color use** - text and accents
- 3 **Hierarchy through typography**
- 4 **Minimal borders and dividers**
- 5 **Clean, professional appearance**

## Benefits

- Maximum readability
- **Reduced visual clutter**
- Professional appearance
- **Better printing**
- Accessibility friendly

**Clean** · Professional · **Effective**