Revolutionizing Corporate Morale with Excel and AI: An End-to-End Pipeline for Automated Birthday Greetings

Antanas Bareikis

January 29, 2025

Abstract

This paper introduces a novel pipeline for automating birthday greetings in the corporate workplace, leveraging state-of-the-art text-to-image models, Python, VBA, and the unsung hero of financial institutions: Microsoft Excel. By combining machine learning innovation with the undeniable ubiquity of Excel in banking operations, this solution delivers personalized birthday greetings rendered as pixel-perfect images in Excel cells. This "groundbreaking" AI-driven approach not only showcases the power of modern technology but also reminds us that, in banking, Excel is the ultimate pillar of operational stability.

1 Introduction

Banks and other financial institutions are often seen as bastions of innovation in fields such as risk analysis, portfolio optimization, and fraud detection. However, let us not mince words: none of this would function without Excel holding it all together. From financial models to project plans, Excel is the duct tape of the corporate world. It is in this spirit that we present our innovative use of Excel for a purpose as monumental as employee morale: automating personalized birthday greetings.

The key contribution of this work lies in transforming cutting-edge AI-generated images into Excel-rendered masterpieces using VBA. This ensures the entire pipeline remains within the sacred confines of Excel, where all banking miracles are born.

2 Architecture

The architecture of the proposed solution consists of the following components:

1. **Large Language Model (LLM) Prompt Generation**: An LLM generates initial text prompts for the image generation step. These prompts are tailored to match the day's horoscopes, adding a layer of personalization to the birthday greeting.

2. **Text-to-Image Model**: Using the Stable Diffusion v1-5 model, personalized birthday images are created based on the generated prompts. The images are generated at a resolution of 512x512 pixels, the resolution the model was trained on.

3. **Image Downscaling and Pixelation**: The generated images are downscaled slightly for manageability and then transformed into a CSV file containing RGB values for each pixel.

4. **Excel Rendering with VBA**: VBA code adjusts Excel cells to create a pixel-like grid by rounding and resizing the cells. Each cell is then colored based on the RGB values from the CSV file.

5. **Automated Email Delivery**: Finally, the rendered Excel file is emailed to the recipient using Python's `smtplib` library.

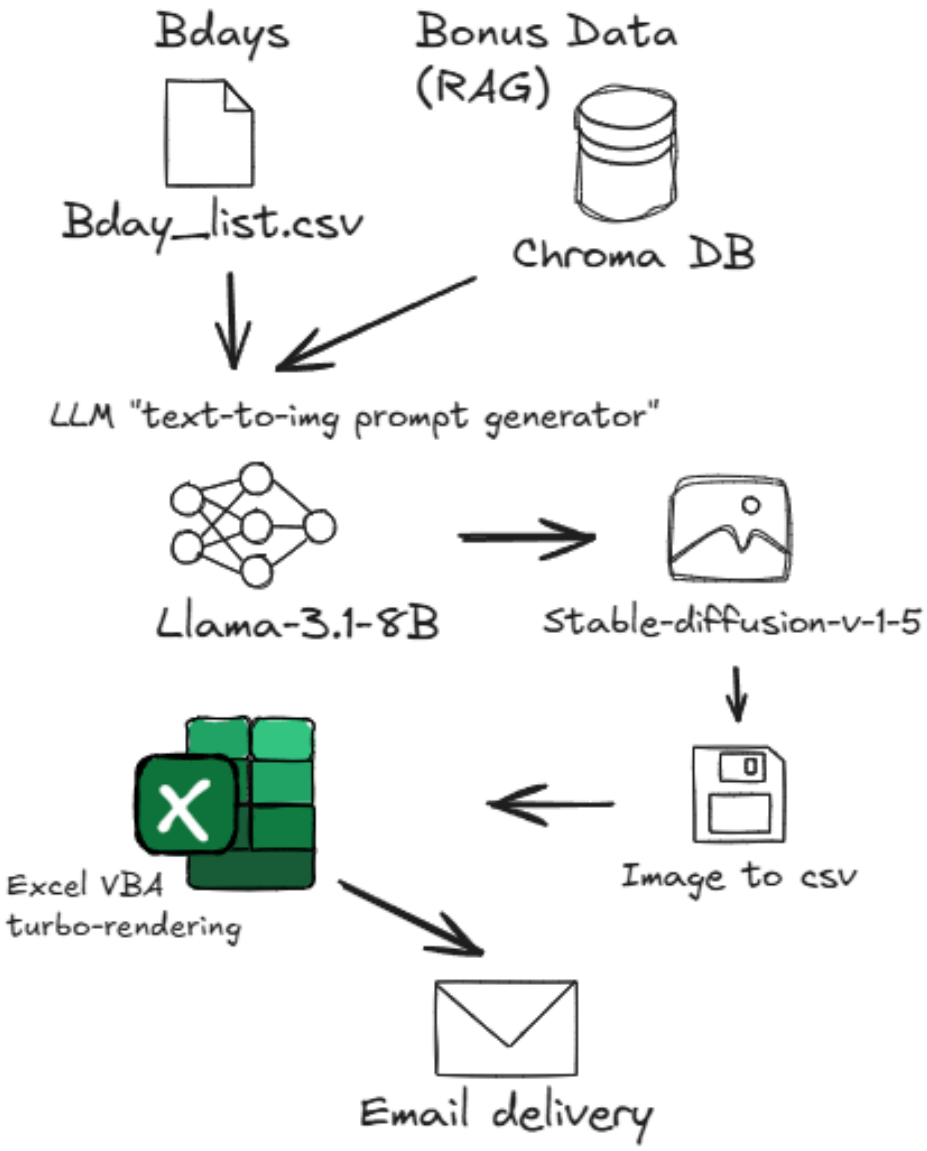


Figure 1: Pipeline Architecture: An end-to-end process from prompt generation to Excel rendering and email delivery.

3 Technical Implementation

3.1 Text-to-Image Model

For the proof of concept, we used the Stable Diffusion v1-5 model, which generates images at a resolution of 512x512 pixels. The choice of this resolution is intentional, as it aligns with the model's training parameters [1], ensuring optimal quality. The input prompts for the text-to-image model are generated from the outputs of LLM (llama3.1-8B) which itself gets inputs from pre-set parameterized prompts which get their dynamic params from our bdaylist.csv.

```
stable_diffusion_1.5/
├── model_index.json
├── README.md
├── feature_extractor/
│   └── preprocessor_config.json
├── safety_checker/
│   ├── config.json
│   └── model.safetensors
├── text_encoder/
│   ├── config.json
│   └── model.safetensors
├── tokenizer/
│   ├── merges.txt
│   ├── special_tokens_map.json
│   ├── tokenizer_config.json
│   └── vocab.json
├── unet/
│   ├── config.json
│   └── diffusion_pytorch_model.safetensors
└── vae/
    ├── config.json
    └── diffusion_pytorch_model.safetensors
```

Figure 2: Stable diffusion v1.5 model structure locally

```python
#loading the stablediffusion model
from diffusers import StableDiffusionPipeline
import torch

model_path = "./stable_diffusion_1.5" # path to your local folder

pipeline = StableDiffusionPipeline.from_pretrained(
    model_path,
    torch_dtype=torch.float16, # or float32, depending on hardware
)
pipeline.to("cuda")
```

3.2 Image to CSV Conversion

Each generated image is downscaled slightly to improve performance and then converted into a CSV file containing RGB values for each pixel. We use NumPy for the initial conversion:

```
    # Convert to NumPy array: shape = (H, W, C)
    img_array = np.array(image)

    # If the image has an alpha channel (RGBA), strip it (keep only RGB).
    if img_array.shape[-1] == 4:
        img_array = img_array[..., :3]
```

And the format of the CSV file is as follows:

```
Row,Col,R,G,B
1,1,255,0,0
1,2,0,255,0
...
```

3.3 Excel Rendering via VBA

Excel's ability to render anything visually appealing—beyond its traditional data visualization and computational use cases—has been demonstrated in highly creative projects. For instance, the classic example of running DOOM within Excel [2], showcasing how versatile this tool can be.

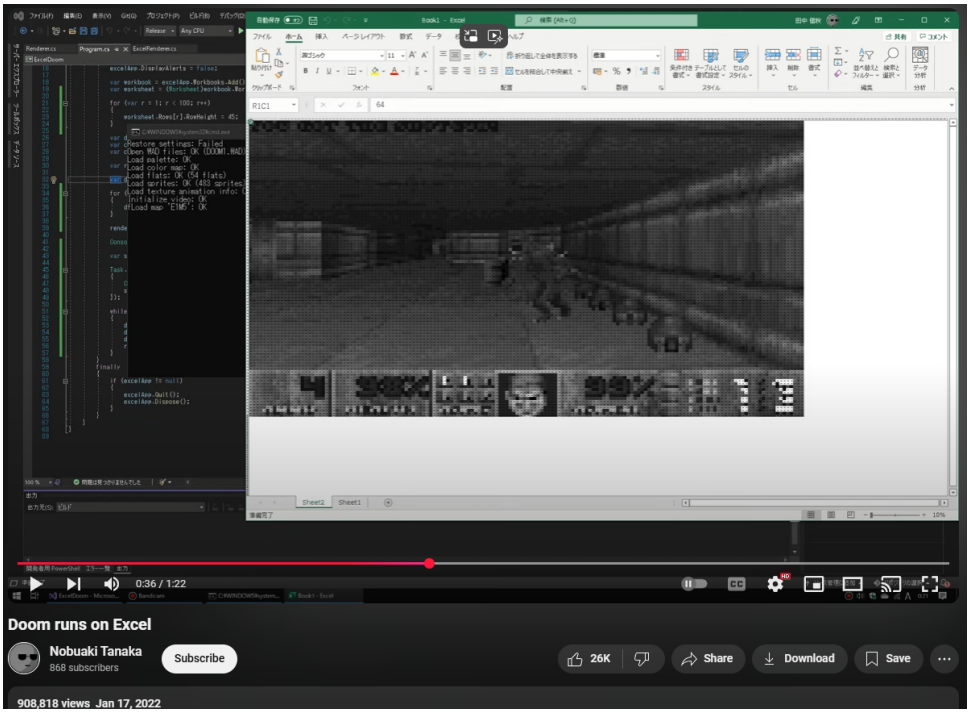


Figure 3: Can it run doom?

Following a similar philosophy, our pipeline uses Excel as a canvas for rendering pixel-perfect images generated through AI. Below is the state-of-art [3] VBA code for rendering images in Excel:

```
    Application.ScreenUpdating = False
    ' ------------------------------
    ' 3. Only blood moves the wheels of history:
    ' ------------------------------
    For i = 1 To rowCount
        For j = 1 To colCount
            ' Set the Interior Color using the RGB values
            newWs.Cells(i, j).Interior.Color = RGB(Rdata(i, j), Gdata(i, j),
                ↪ Bdata(i, j))
        Next j
    Next i

    Dim colWidth As Double
    Dim rowHeight As Double
    colWidth = 0.2
    rowHeight = 1.5 'excel uses different metrics for width and height for some
        ↪  reasons beyond my comprehension (???)
```

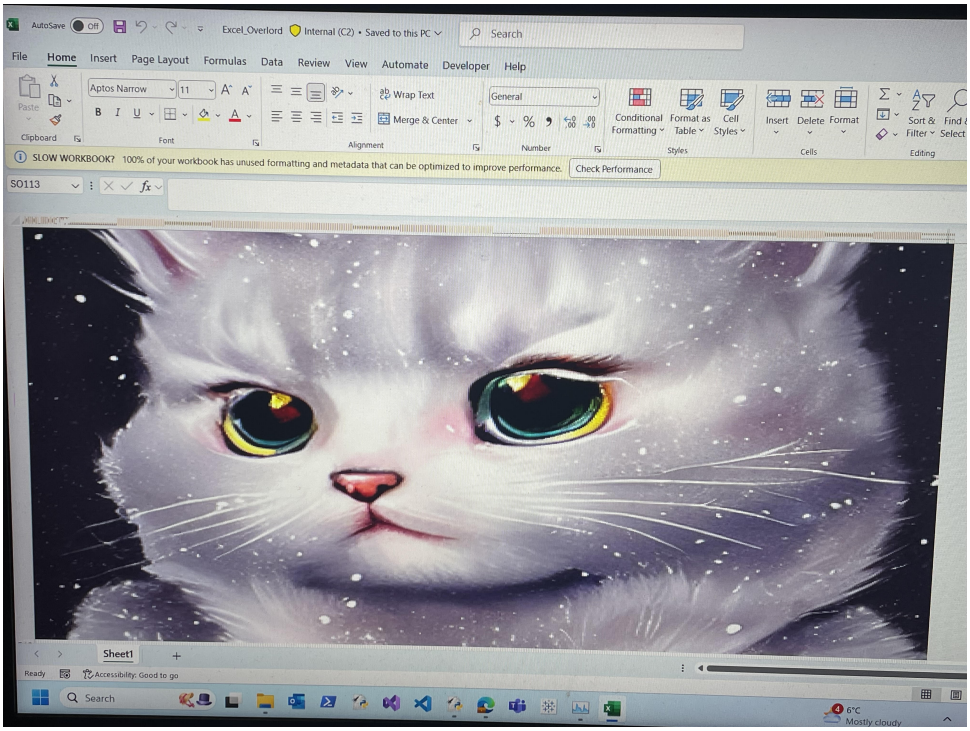


Figure 4: State-of-the-Art Ultra-High-Definition Feline Rendering (Excel Edition)

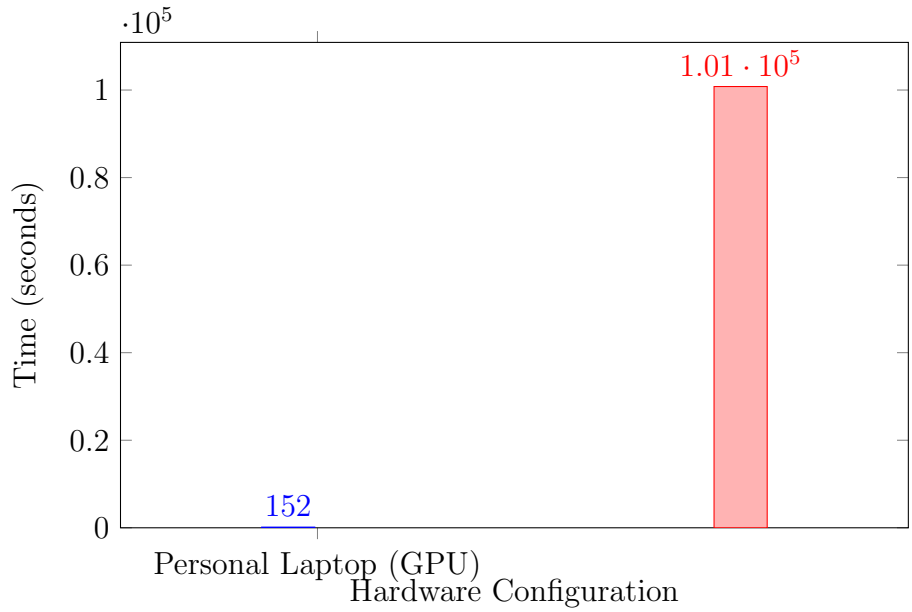
4 Performance Tests

We evaluated the pipeline's performance on two hardware configurations:

- **Casual Personal Laptop**: Equipped with an NVIDIA RTX 4070 GPU, the entire pipeline ran with `device="cuda"` in **2 minutes and 32 seconds (0:02:32)** for a 512x512 image.
- **Corporate Laptop**: Lacking a GPU, the pipeline ran on a 13th Gen Intel® Core™ i9-13950HX CPU (24 cores). For a 128x128 image, the pipeline took **1 hour, 48 minutes, and 49 seconds (1:48:49)**. Extrapolating this performance for a 512x512 image suggests a time of:

$$T_{512} = T_{128} \times \left(\frac{512}{128}\right)^2 = 1:48:49 \times 16 \approx 28\,\text{hours} \tag{1}$$

4.1 Chart of Performance Results



Based on these findings, we humbly suggest that the bank should consider upgrading its hardware. After all, if Excel is the glue holding the financial world together, then GPUs are the glitter that can make it sparkle.

5 Results

The solution has been deployed successfully in a corporate banking environment. Test recipients have universally expressed "appreciation," some even referring to the Excel-rendered images as "art." One particularly inspired colleague remarked, "This is exactly the kind of innovation our industry needs."

6 Discussion

While some may question the necessity of rendering images in Excel, the choice was deliberate. Excel's ubiquity ensures the solution is accessible to all employees without requiring additional software installations.

7 Conclusion

This work demonstrates that the combination of cutting-edge AI and the unyielding stability of Excel can yield innovative solutions for even the most mundane corporate tasks. We invite others to build upon this revolutionary approach, perhaps extending it to other morale-boosting applications.

Acknowledgments

We thank Excel for its unwavering service to the banking sector and our colleagues for tolerating this experiment.

References

[1] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, B. Ommer. High-Resolution Image Synthesis with Latent Diffusion Models. *arXiv preprint arXiv:2112.10752*, 2022.

[2] "DOOM running in Excel," YouTube, `https://www.youtube.com/watch?v=J2qU7t6Jmfw`.

[3] This was once revealed to me in a dream.