A. Dataset Details

A.1. Tire Development Process

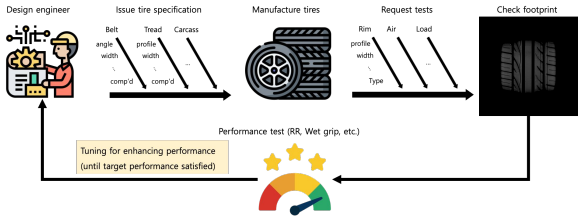

Figure 6. Illustration of the tire development process. An engineer designs tire specifications, and a tire is manufactured to those specifications. For the manufactured tire, photographs are taken of the area where the tire touches the ground under various test conditions, which is called the tire footprint image. Our goal is to generate a tire footprint image directly from the tire specification and test conditions without manufacturing the tire.

In the tire manufacturing industry, the development of new tires is both labor-intensive and resource-demanding. Initially, engineers design a tire by detailing various technical specifications, which guide the creation of experimental prototypes. A crucial step in evaluating these prototypes involves capturing images of the tire's contact patch with the ground, referred to as tire footprints, under diverse conditions. These footprints are essential for assessing the tire's performance and for engineers to visually verify that the prototype matches their design intentions. Often, this evaluation leads to further adjustments in tire specifications. Fig. 6 shows the process for developing a tire.

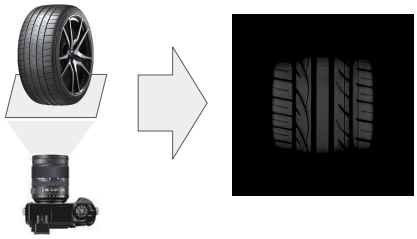

Figure 7. An illustration of tire footprint

Tire Footprint plays a very important role in the tire development process. Tire developers look at the Tire Footprint to not only determine if their design is working as intended, but also to quantify the image to evaluate the performance of the tire. Here are three of the tire performance metrics we used in our experiments. Figure 7 illustrates the description of the tire footprint image, and Figure 8 shows an example of a tire footprint.

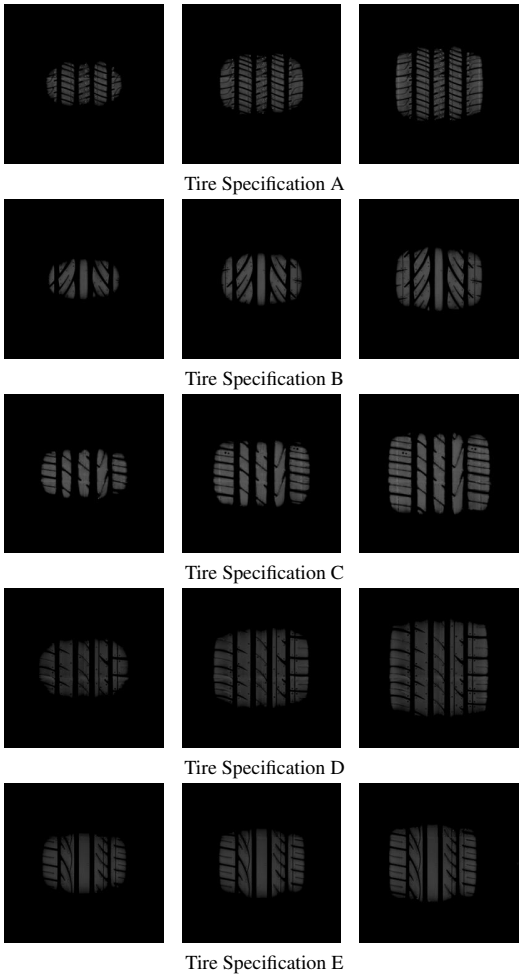

Figure 8. Ground truth images for increasing test load, $\rightarrow$ in order. As the test load increases, the contact area increases.

A.2. Example of Tabular Data

Table 4 provides examples of tabular data. With 116 features in total, it is impractical to show all of them, so we highlight key examples. For instance, the tire specification features include "Pattern" (a categorical feature) and "Tire Size" (a numeric feature). Additionally, graphical features such as "CTB_Step_Width" and "CTB_Step_Gauge" denote position and value, respectively, forming a graph when connected. Test condition features include "Test_Load" and "Inflation_air," where "Test_Load" indicates the load applied to the tire (affecting the contact area) and "Inflation_air" denotes the tire's air pressure. These two test conditions are the main conditions for tire development, and the ground contact area of the tire will increase or decrease depending on the change in test conditions.

Table 4. Examples of tabular data

Feature Name	Tire Spec or Test Condition	Type	Example
Pattern	Tire Specification	Category	H826
Tire Size	Tire Specification	Numeric	250
...	...	...	...
CTB_Step_Width	Tire Specification	Graph	5.0-3.0-7.0-6.0-...
CTB_Step_Gauge	Tire Specification	Graph	0.0-0.2-0.2-0.2-...
...	...	...	...
Inflation_air	Test Condition	Numeric	3.4
Test_load	Test Condition	Numeric	256

A.3. Example of Graphical Feature

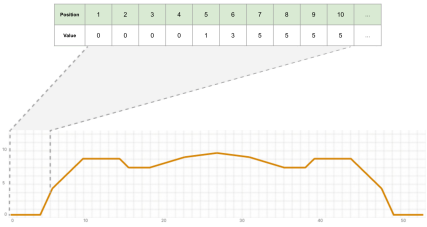

Figure 9. Illustration for graphical feature

A Graphical Feature refers to the structure of a specific part used in a tire. Tabular data is recorded in two categories: position and value. We visualize it and generate an embedding using a CNN. This data preprocessing approach demonstrates the scalability of tabular-to-image conversion for multi-modal applications. Figure 9 illustrates the example of graphical feature.

B. Implementation Details

B.1. Latent Diffusion U-net

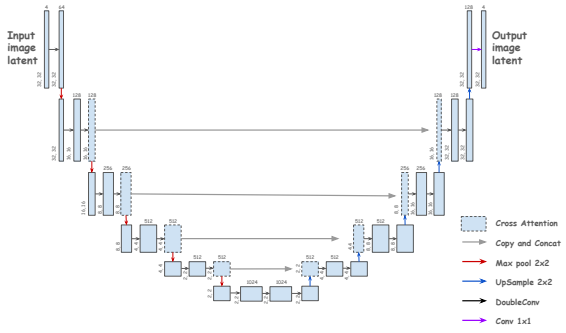

Figure 10. Illustration of the U-net structure.

The structure of the U-net we use is depicted in Figure 10. It consists of five downsampling and five upsampling steps, projecting the embedding into latent space using a VAE. In the encoder part, which performs the downsampling, we apply cross-attention on the embedding before downsampling. Similarly, in the decoder part, which handles the upsampling, cross-attention is applied after upsampling. We condition the embedding created by the Tabular encoder by performing cross-attention on a layer of the U-Net.

B.2. Transform Tabular Data into Natural Language

In Experiment 4.4, we also examined the results of using natural language models to convert tabular data into natural language. We transformed the tabular data into natural language, except for graphical features, which we converted into natural language for feature names and feature values. Below are some examples of transforming tabular data into natural language:

Pattern: H825
Tire Size: 250
.
.
Inflation air: 3.4
Test Load: 256

The graphical features were generated as embeddings by a CNN and concatenated with embeddings generated by a natural language model. This experiment involved replacing the tabular encoder in the CTIP framework with an encoder that solely utilizes a natural language model.

B.3. Hyperparameter Settings

First, we pre-train on CTIP. CTIP will be trained with a total of 30 epochs and will only be trained with the train dataset. We freeze the weights of the tabular encoder on the pre-trained CTIP and then train DF-GAN and LDM on the pre-trained CTIP. DF-GAN is trained for a total of 100 epochs, and LDM is trained for a total of 1000 epochs. Relatively speaking, LDM is faster to train, taking about the same amount of time (36 hours). In the case of LDM, VAE

Table 5. CTIP hyperparameter setting

	CTIP	DF-GAN	LDM
Batch size	128	128	128
Embedding dimension	256	256	256
Learning rate	0.001	0.0001	0.0001
Training epochs	30	100	1000
Adam β_1	0.0	0.0	0.0
Adam β_2	0.9	0.9	0.9
Temperature t	0.01	-	-
ViT layers	4	-	-
ViT heads	4	-	-
ViT patch size	4	-	-

imports and uses pre-trained weights.

C. Experiment Details

C.1. Tire Performance Metrics

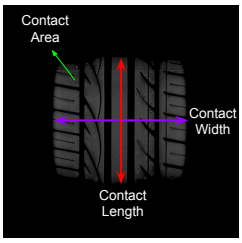

Figure 11. Illustration of tire performance metrics.

The Tire Performance metrics CL, CW, and CA are calculated based on the contact area of the tire. To measure these, we first remove the background of the generated image, retaining only the tire footprint—defined as the area where the tire has made contact—and set the remaining pixel values to zero. CL is determined by measuring the distance from the topmost to the bottommost non-zero pixel within the tire footprint. CW is measured from the leftmost to the rightmost non-zero pixel within the tire footprint. CA is calculated as the total number of non-zero pixels, representing the area touched by the tire. These metrics are crucial for evaluating tire performance and are used to rate tires based on specific criteria. Figure 11 shows an example of a tire performance metric.

C.2. Image Results

Figures 12 and 13 showcase the results of experiments using a tabular encoder and a BERT encoder, respectively. These figures display the outcomes for three test conditions with two tire specifications and the resulting images generated by both DF-GAN and LDM with and without pre-trained models. Overall, the tabular encoder produced more accurate images compared to using BERT with text conversion of the tabular data. Although pre-training with CLIP after generating embeddings with BERT by converting the tabular data to text led to more accurate images, they were not as precise as those produced by pre-training the tabular encoder with CTIP.

On the other hand, using a tabular encoder and pre-training it with CTIP resulted in nearly accurate tire footprints. Improvements were observed for both DF-GAN and LDM, with LDM showing the best performance when pre-trained with CTIP. It is evident that this approach accurately captures the geometry of the tire footprint, which is crucial for reliable image generation.

C.3. Few-shot and Zero-shot Image Results

Figure 14 shows the result images for feature few-shot and zero-shot. Without CTIP, the generated images look significantly different from the GT images and tend to randomly produce certain patterns, which is a poor result as it generates incorrect images. In contrast, when CTIP is used, the tire footprint shape is generated more accurately. Notably, in the first pattern case, feature zero-shot, the overall shape is roughly correct. Additionally, with CTIP, unseen patterns appear blurred rather than being randomly generated. These results demonstrate that CTIP is effective in generating robust images, even in challenging cases.

C.4. Pesudo Code for Evaluating CTIP through Image Similarity

Algorithm 1 presents the pseudocode for the experiment detailed in Analysis 5.3. To validate that CTIP indeed produces high-quality embeddings, we conducted an experiment to determine if the similarity between real images and the embeddings from a tabular encoder pre-trained with CTIP are closely aligned. Initially, we compute the LPIPS (Learned Perceptual Image Patch Similarity) between the ground truth images of the test set to create an $N \times N$ matrix S_{LPIPS} (steps 3-11 in Algorithm 1). Subsequently, we generate embeddings for the test set from both the CTIP-trained and untrained tabular encoders. We then measure the Euclidean distance between these embeddings to create the $N \times N$ matrices $D_{\text{with CTIP}}$ and $D_{\text{without CTIP}}$ (steps 12-23 in Algorithm 1). Finally, we compute the cosine similarity of $D_{\text{with CTIP}}$ and $D_{\text{without CTIP}}$ against S_{LPIPS}. By comparing the two cosine similarities, we analyze the results as shown in Analysis 5.3.

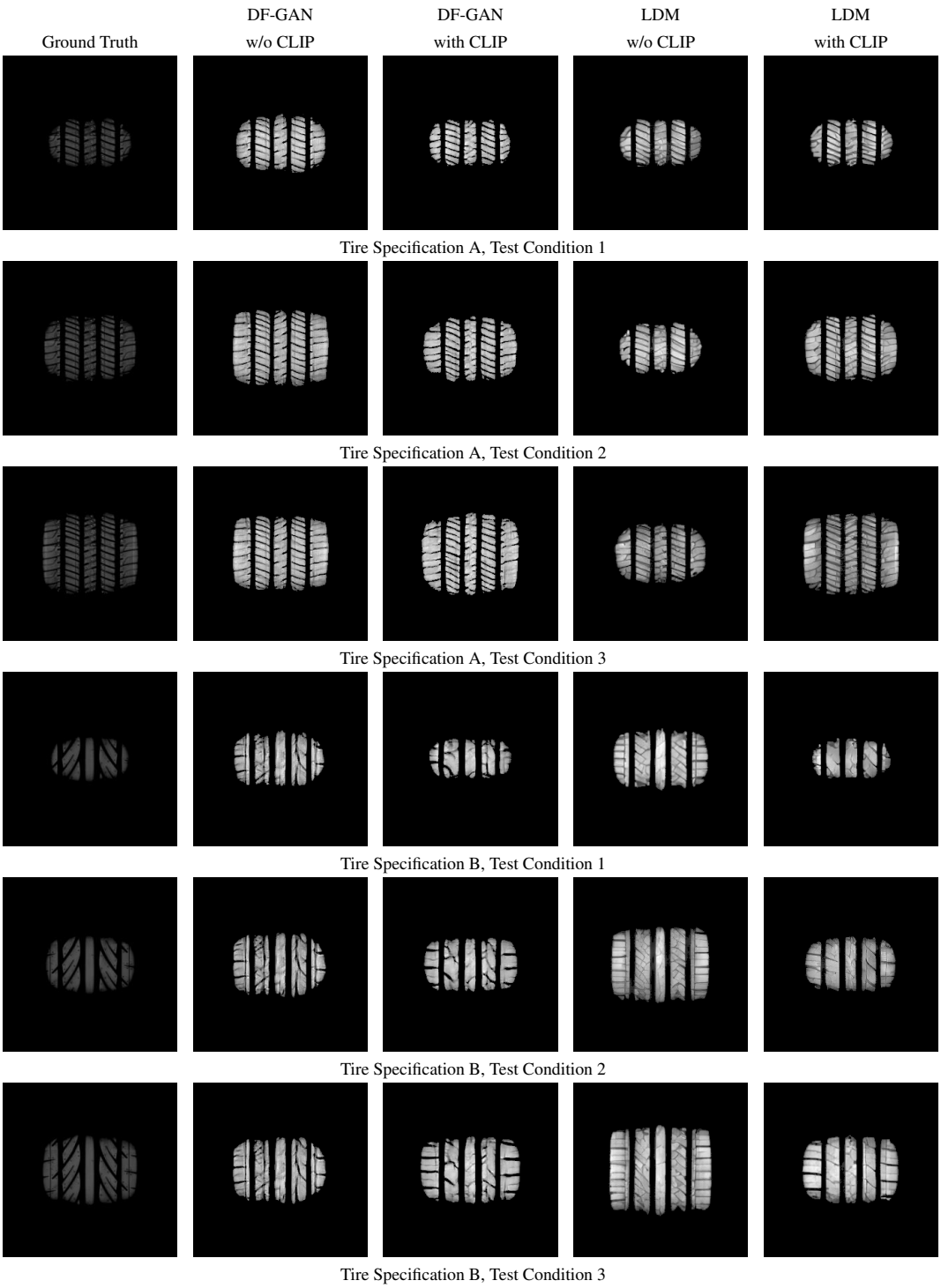

Figure 12. Examples of tire footprint images for three test conditions using CTIP.

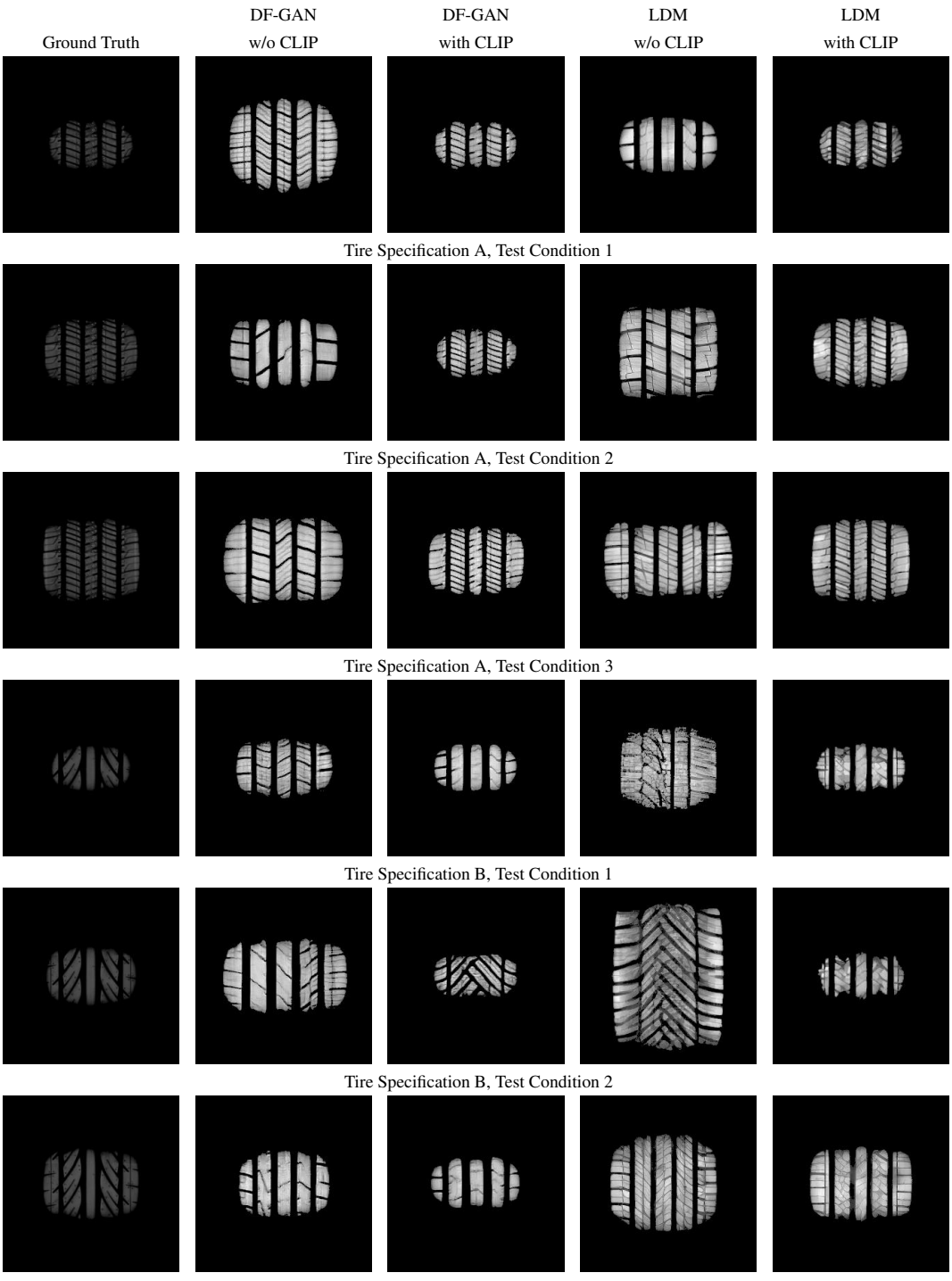

Figure 13. Examples of tire footprint images for three test conditions using clip.

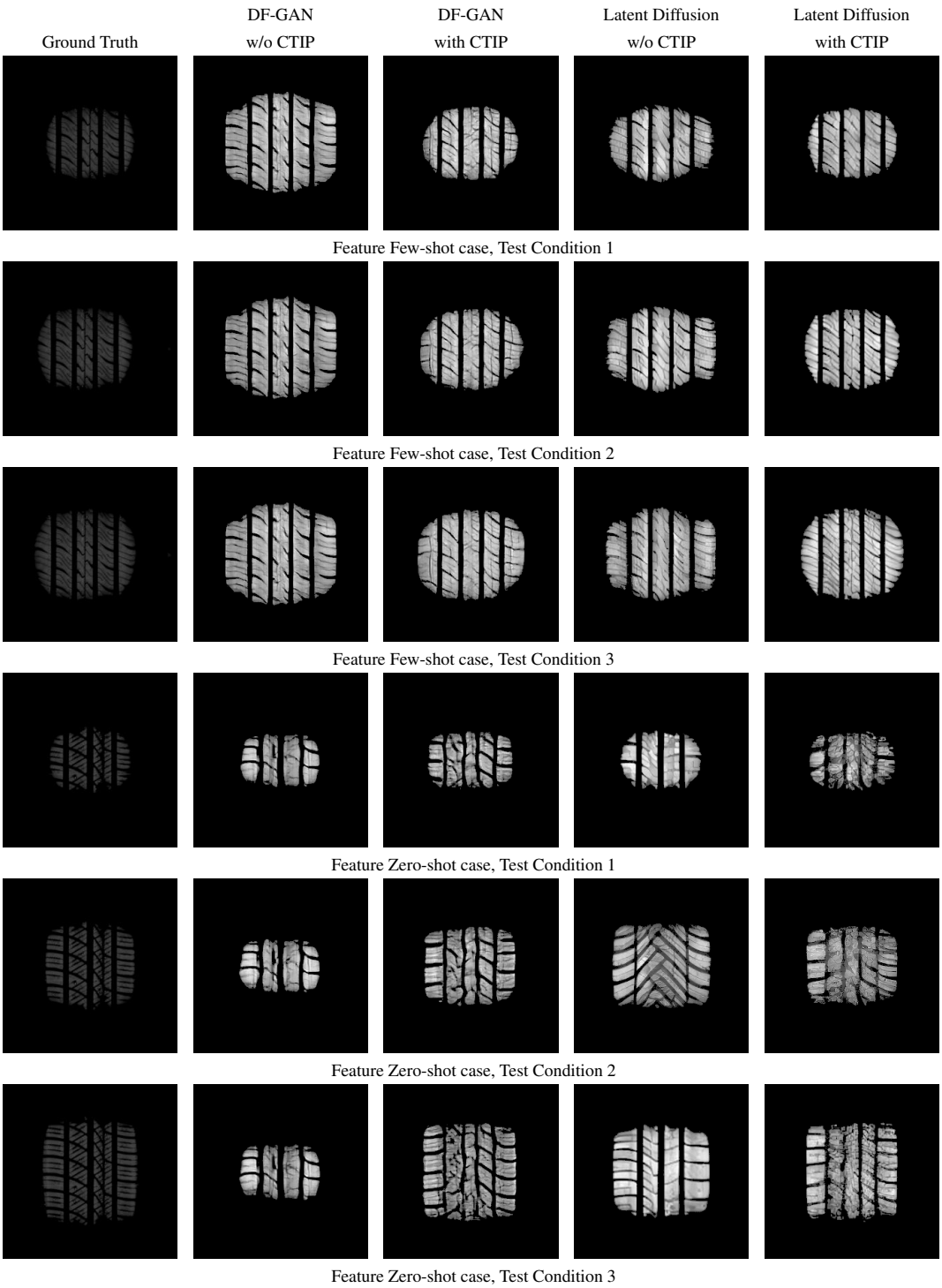

Figure 14. Examples of the feature few-shot & zero-shot cases, where there is under five tire specifications with the same tire pattern in the train dataset. The zero-shot case is that there are no tire specifications with the same tire pattern in the train dataset.

Algorithm 1 Pesudo code for Evaluating CTIP through Image Similarity

Input: gt images I_{gt}, condition embeddings with CTIP $E_{\text{with CTIP}}$, condition embeddings without CTIP $E_{\text{without CTIP}}$
Output: Cosine Similarity
procedure IMAGEMATRIX(I_{gt})
    $N \leftarrow$ Numer of test sets
    Initialize S_{LPIPS} as a $N \times N$ matrix
    for each pair of images in I_{gt} **do**
        Compute LPIPS similarity
        Update S_{LPIPS}
    end for
    return S_{LPIPS}
end procedure
procedure DISTANCEMATRIX($C_{\text{with CTIP}}, C_{\text{without CTIP}}$)
    $N \leftarrow$ Numer of test sets
    Initialize $D_{\text{with CTIP}}$ as a $N \times N$ matrix
    Initialize $D_{\text{without CTIP}}$ as a $N \times N$ matrix
    for $C_{\text{with CTIP}}, C_{\text{without CTIP}}$ **do**
        for each pair of embeddings **do**
            Calculate Euclidean distance
            Update $D_{\text{with CTIP}}$ or $D_{\text{without CTIP}}$
        end for
    end for
    return $D_{\text{with CTIP}}$, $D_{\text{without CTIP}}$
end procedure
procedure COMPUTE COSINE SIMILARITY($D_{\text{with CTIP}}$, $D_{\text{without CTIP}}$, S_{LPIPS})
    Calculate cosine similarity between S_{LPIPS} and $D_{\text{with CTIP}}$
    Calculate cosine similarity between S_{LPIPS} and $D_{\text{without CTIP}}$
end procedure
$S_{\text{LPIPS}} \leftarrow$ IMAGEMATRIX(I_{gt})
$D_{\text{with CTIP}}$, $D_{\text{without CTIP}} \leftarrow$ DISTANCEMATRIX($C_{\text{with CTIP}}, C_{\text{without CTIP}}$)
COMPUTECOSINESIMILARITY($D_{\text{with CTIP}}$, $D_{\text{without CTIP}}$, S_{LPIPS})