I Control of the second second

Generation of Synthetic Datasets in the Context of Computer Networks using Generative Adversarial Networks

Thiago Caproni Tavares (IFSULDEMINAS) Ariel Góes de Castro (UNICAMP) Leandro C. de Almeida (IFPB) Washington Rodrigo Dias da Silva (UFSCar) Christian Esteve Rothenberg (UNICAMP) Fábio Luciano Verdi (UFSCar)



Support











Summary

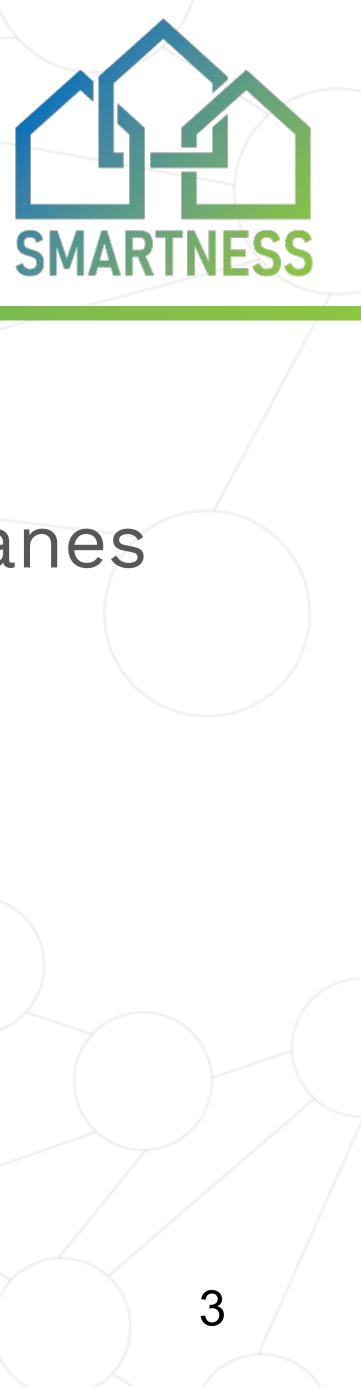
1. Introduction

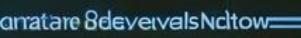
- 2. Fundamentals of Generative Adversarial Networks
- 3. In-band Network Telemetry and Programmable Data Planes 4. Generation of Telemetry Data

a. Hands on

5. Generation of Synthetic Network Trace a. Hands on

6. Conclusions and future perspectives





enternatia do verrial Ne

Introduction







Motivation

Introduction

Why are datasets important for computer networks?

To understand applications with different demands (e.g., latency, throughput) and propose solutions (e.g, protocols)

What are the real/existing dataset limitations? Little data available and privacy of companies and users is affected

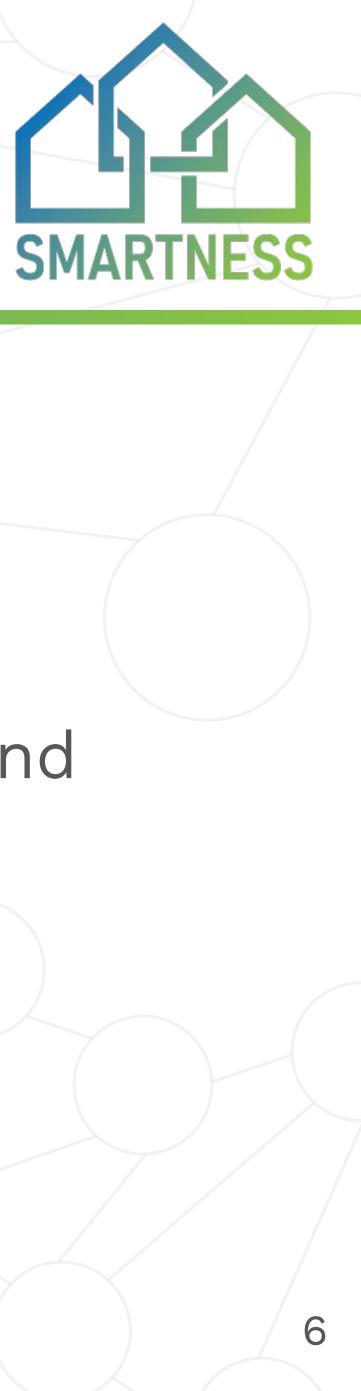
Why use synthetic data in computer networks? Existing datasets may be scarce or outdated - i.e., do not reflect existing applications' needs (e.g., TSN, 5G and beyond, video streaming)



Practical Applications of GANs Introduction

- Generate synthetic data for machine learning model inputs
- Allocation for prediction or classification tasks
- Application in network simulations to maintain data privacy and enhance data quality





Introduction to Generative Artificial Intelligence

Introduction

Generative Artificial Intelligence: An Overview

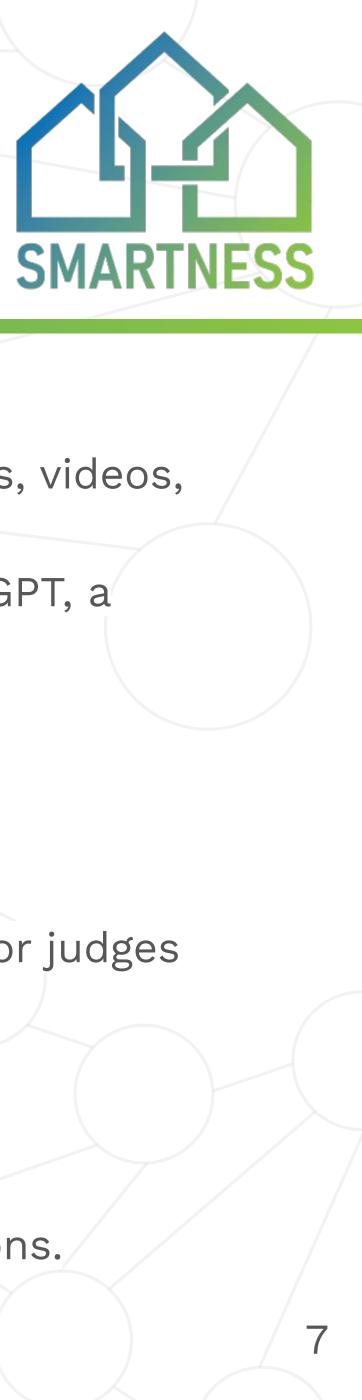
- Encompasses algorithms and models capable of generating diverse data forms, including images, videos, text, and digital media.
- Has gained significant traction outside academic circles, largely due to advancements like ChatGPT, a Large Language Model (LLM).

Generative Adversarial Networks (GANs)

- Introduced primarily for image synthesis.
- Comprises two neural networks: the generator and the discriminator.
- Operates on game theory principles: the generator creates synthetic data, while the discriminator judges its authenticity.

Applications in Computer Networks

- Extension of GAN applications in computer networks, focusing on data synthesis and privacy. • Utilization includes dataset augmentation, balancing, and simulation of complex data distributions.



So... can we generate synthetic network data?





Spoiler: yes





Fundamentals of Generative Adversarial Networks

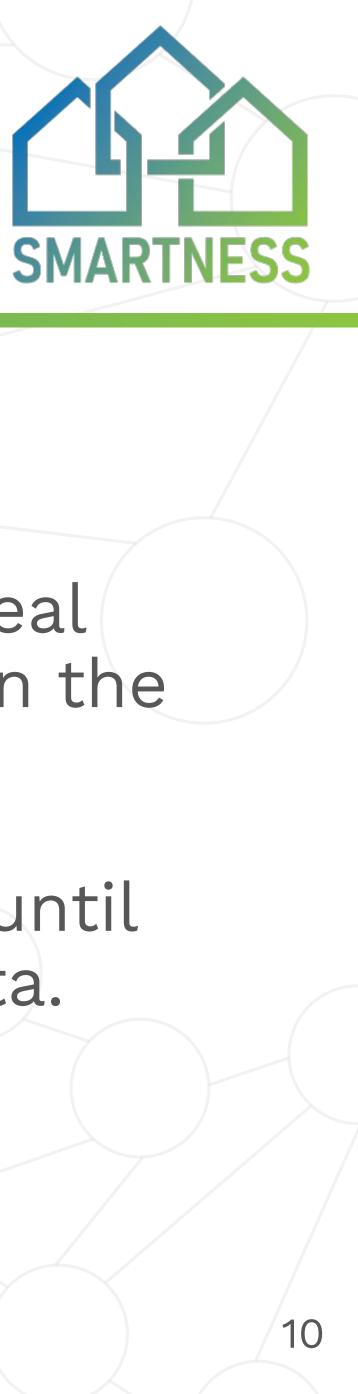




9

Fundamentals of Generative Adversarial Networks

- **Definition:** Machine learning framework consisting of two competing modules: the generator and the discriminator.
- two.
- Complexity: The discriminator can be seen as employing supervised learning by using real data to train its judgment capabilities.

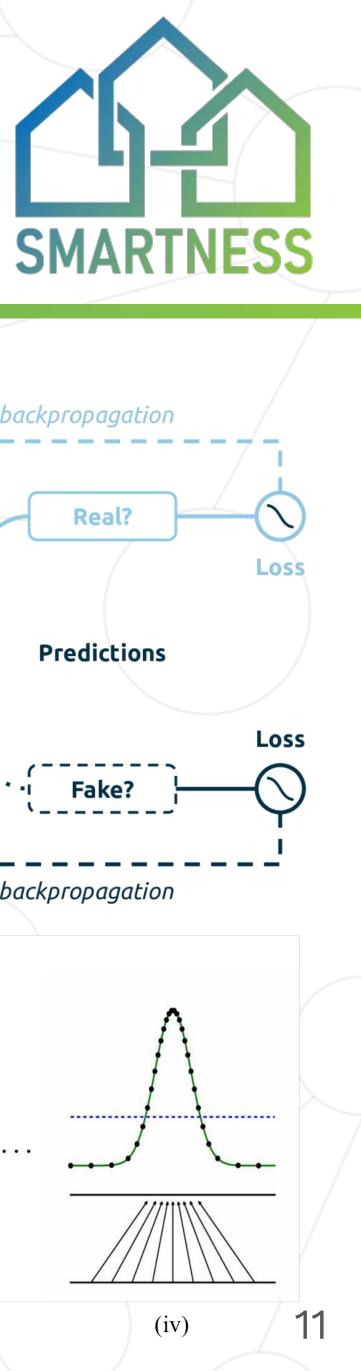


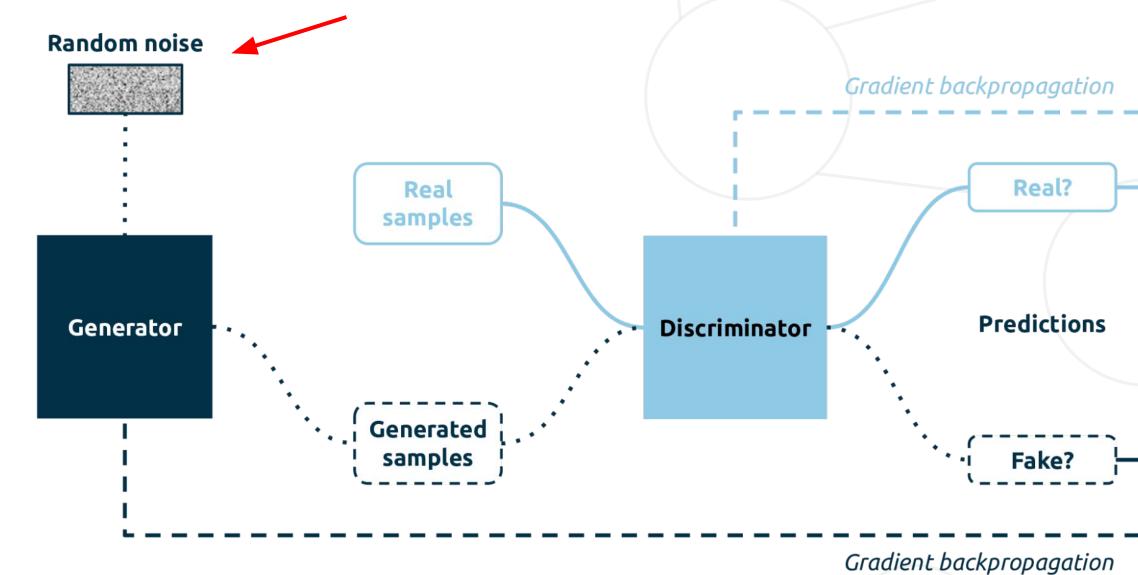
• Function: The generator creates synthetic data mimicking real data, while the discriminator learns to differentiate between the

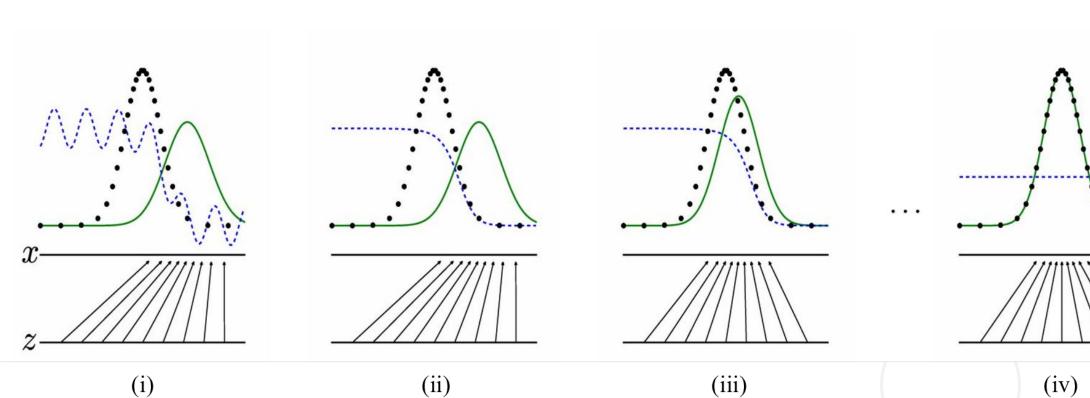
• Training Dynamics: Both modules are trained adversarially until the discriminator cannot distinguish real from synthetic data.

Fundamentals of GANs

- We summarize the GAN training as follows:
- 1. First, we feed the generator with (i) noise and a (ii) training sample

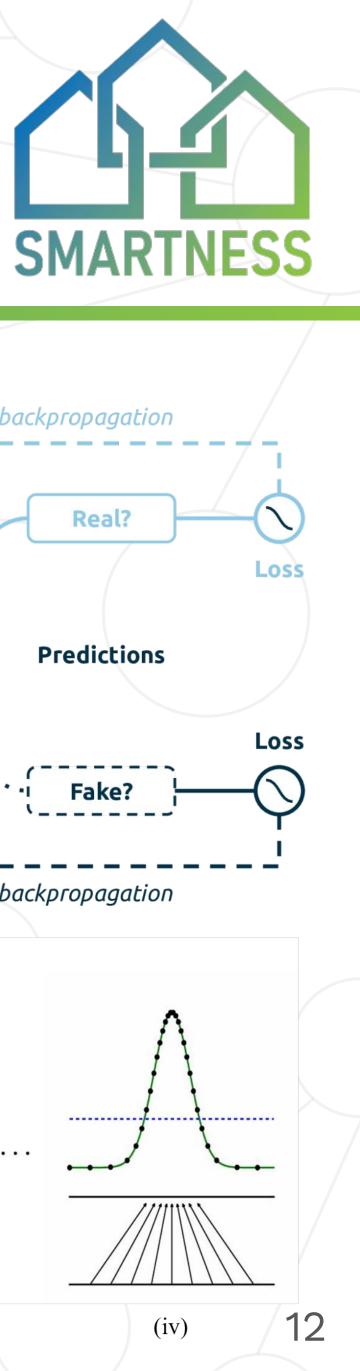


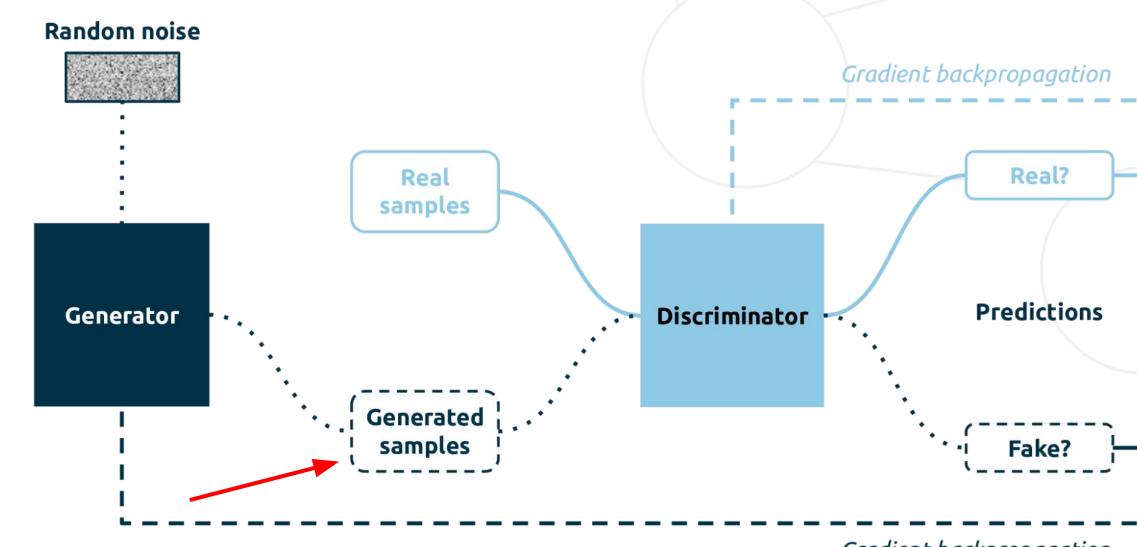




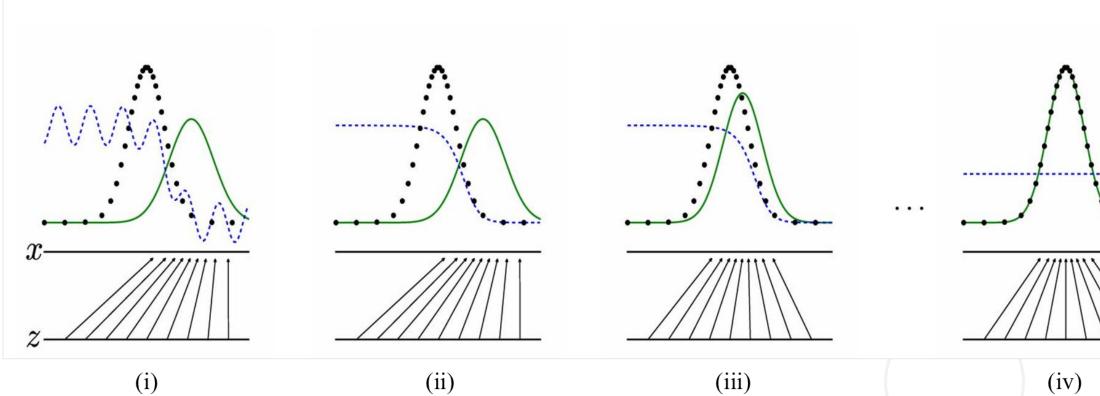
Fundamentals of GANs

- We summarize the GAN training as follows:
- 1. First, we feed the generator with (i) noise and a (ii) training sample
- 2. We merge it and generate a new sample



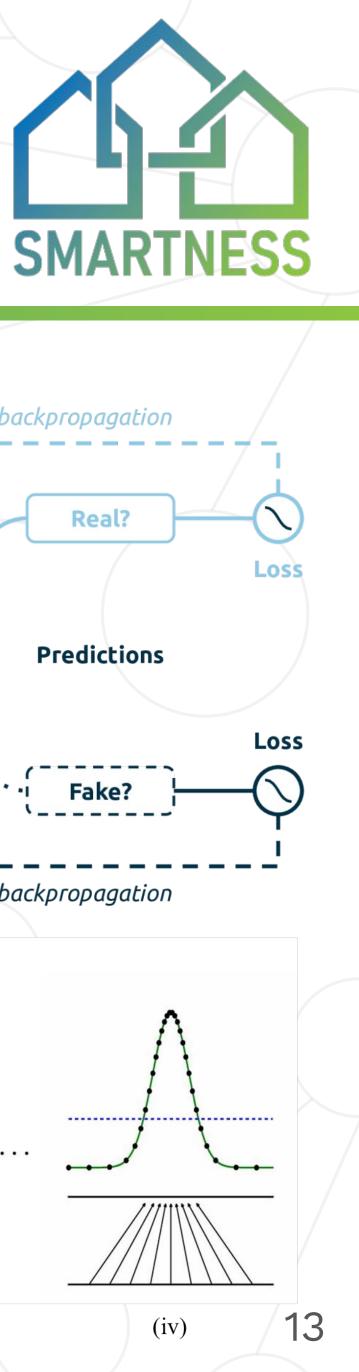


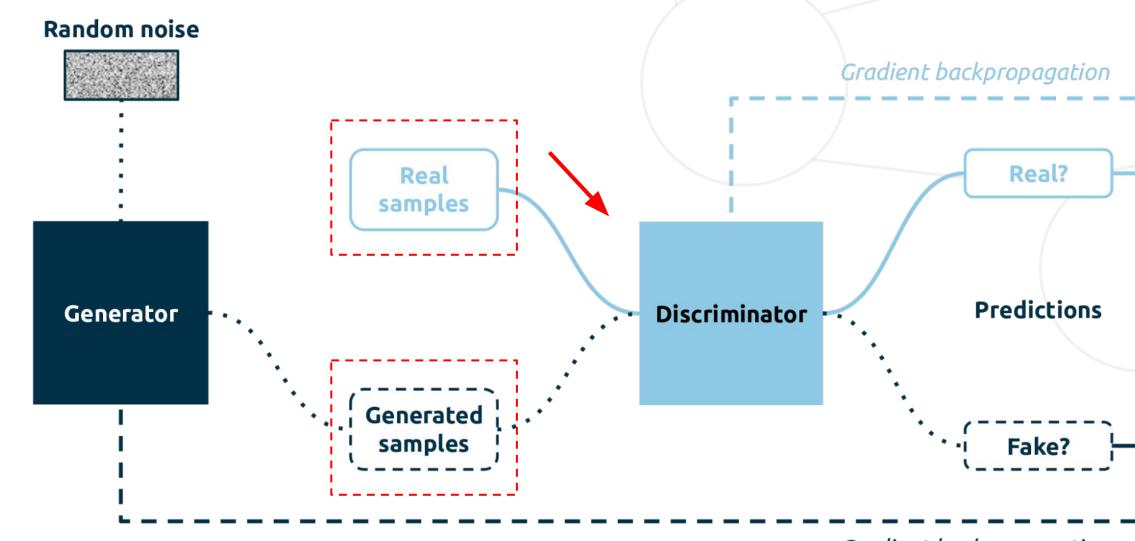
Gradient backpropagation



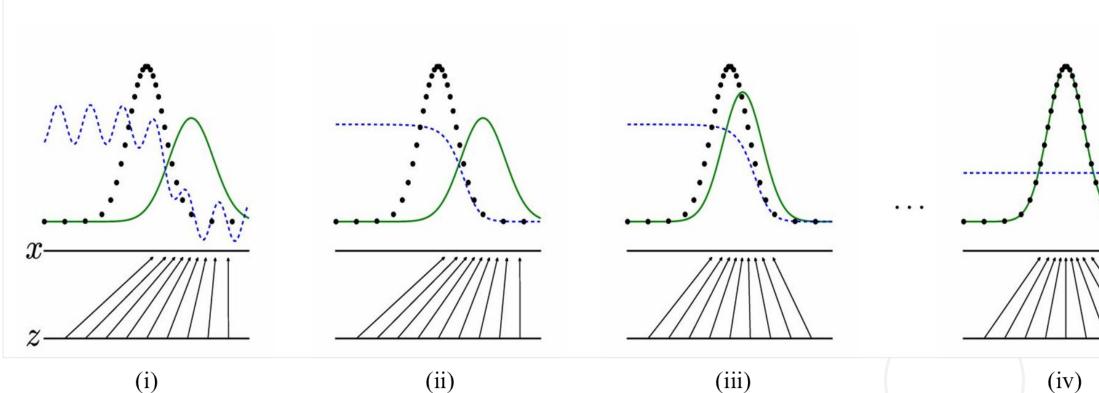
Fundamentals of GANs

- 1. First, we feed the generator with (i) noise and a (ii) training sample
- 2. We merge it and generate a new sample
- 3. The Discriminator tries to guess which of the entries is the real one



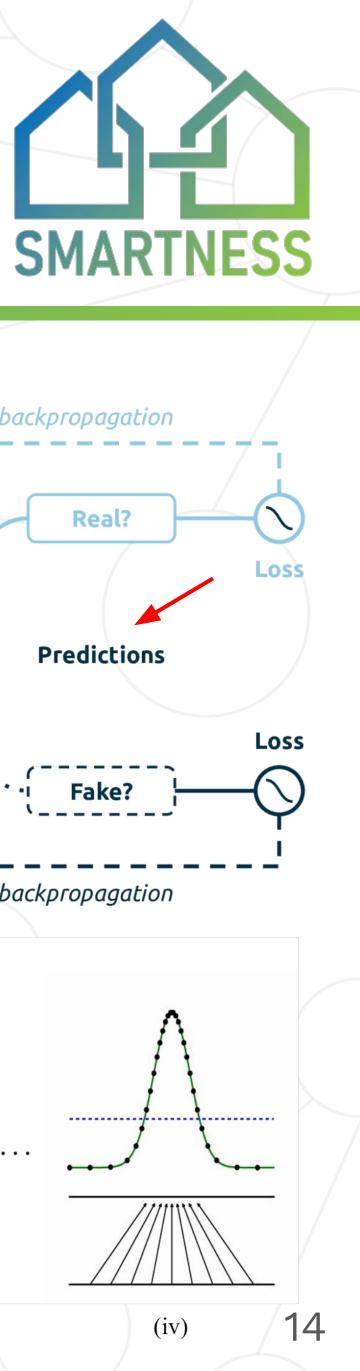


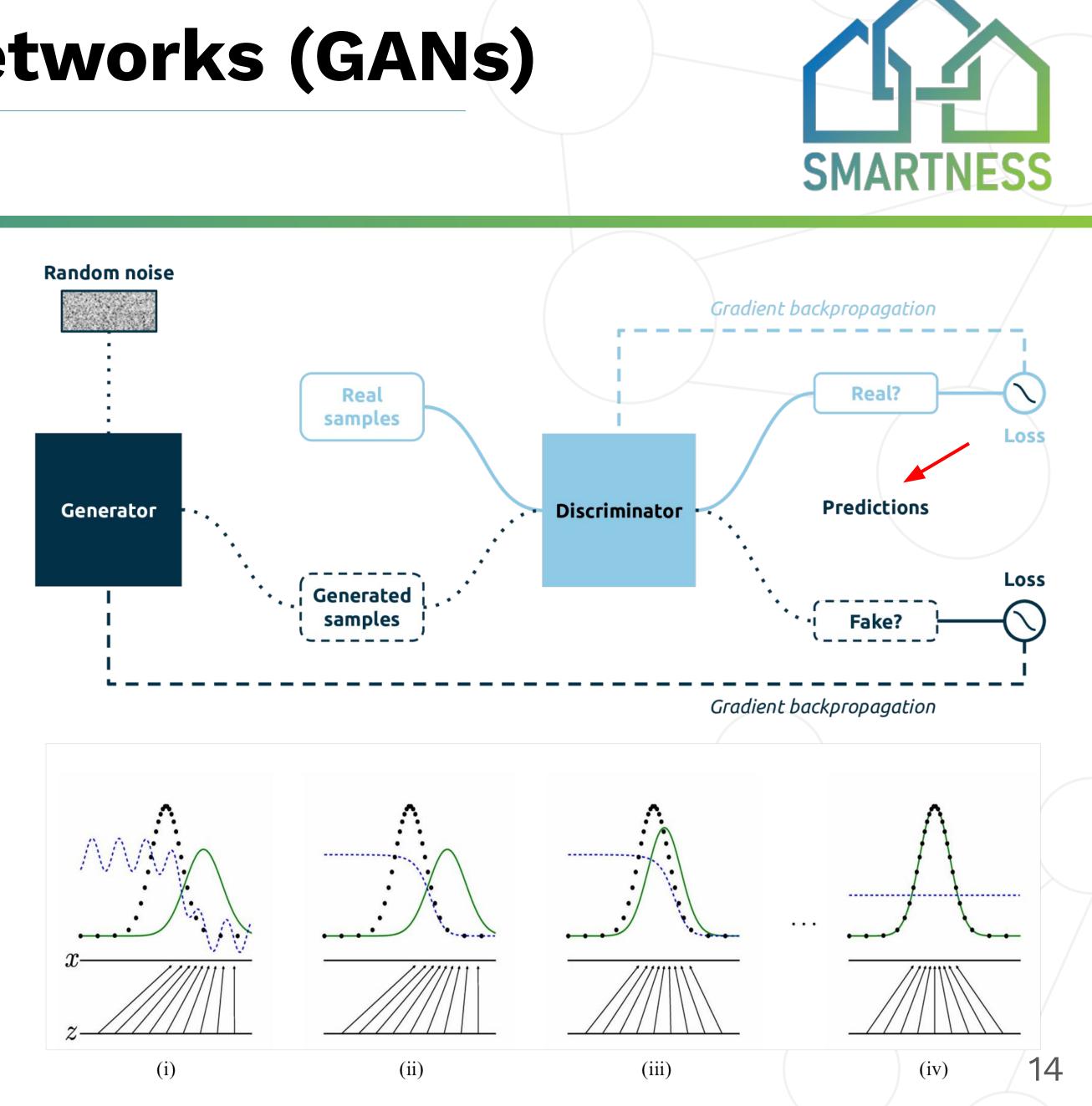
Gradient backpropagation



Fundamentals of GANs

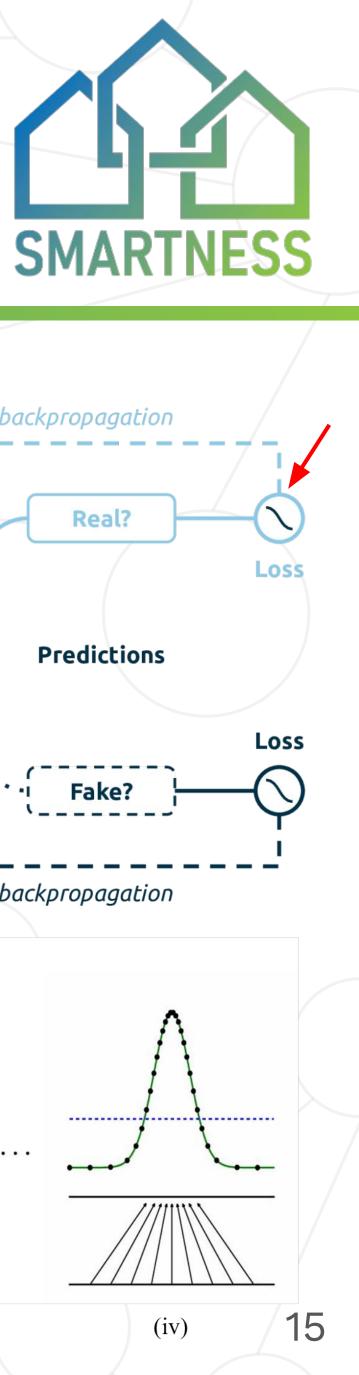
- 1. First, we feed the generator with (i) noise and a (ii) training sample
- 2. We merge it and generate a new sample
- 3. The Discriminator tries to guess which of the entries is the real one
- 4. The prediction is generally in the probability interval [0,1] (e.g., sigmoid function)

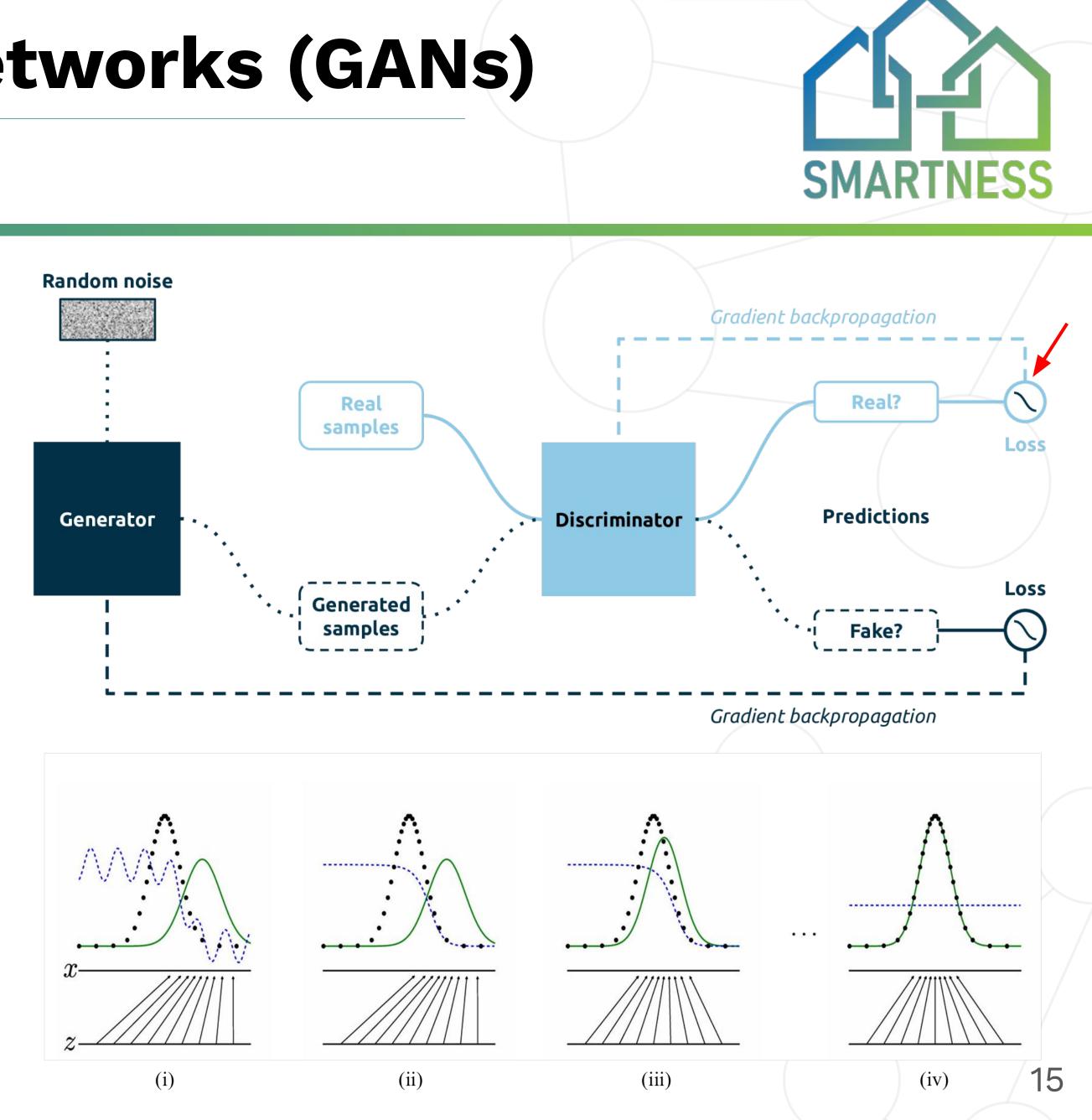


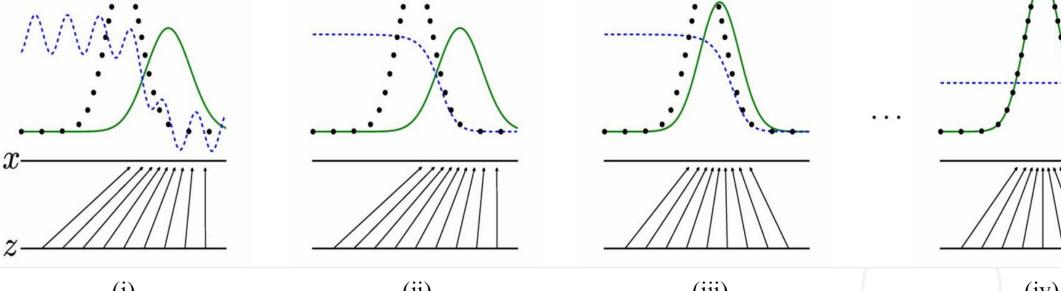


Fundamentals of GANs

- 1. First, we feed the generator with (i) noise and a (ii) training sample
- 2. We merge it and generate a new sample
- 3. The Discriminator tries to guess which of the entries is the real one
- 4. The prediction is generally in the probability interval [0,1] (e.g., sigmoid function)
- 5. The generator is rewarded if it "fools" the discriminator

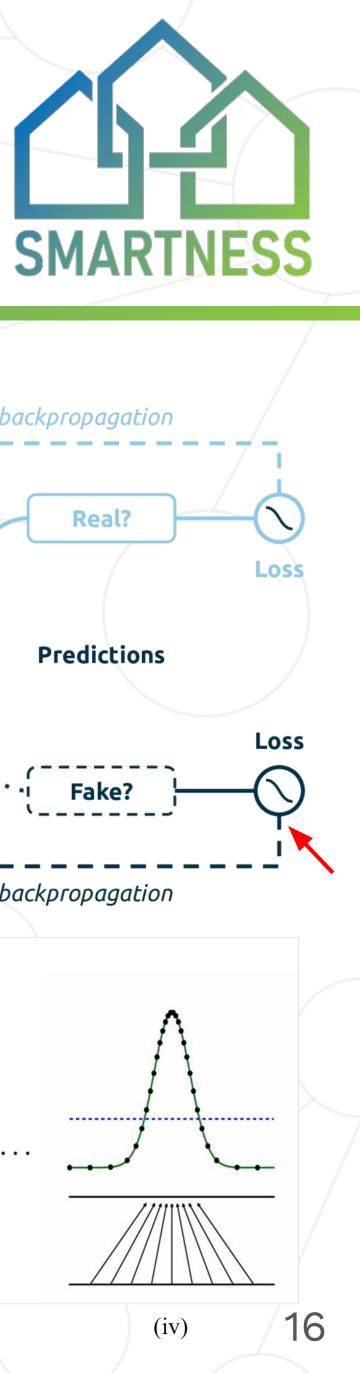


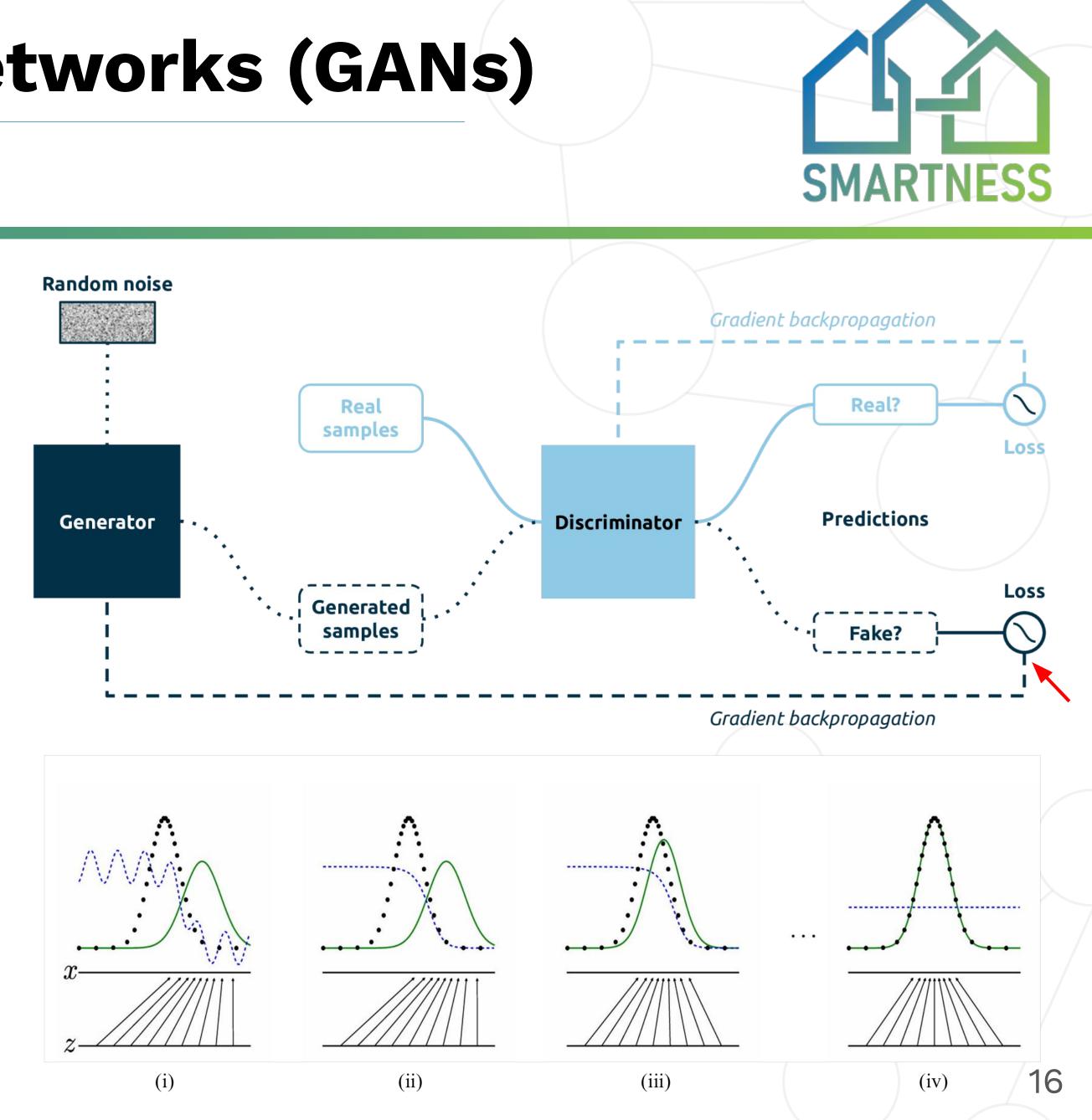


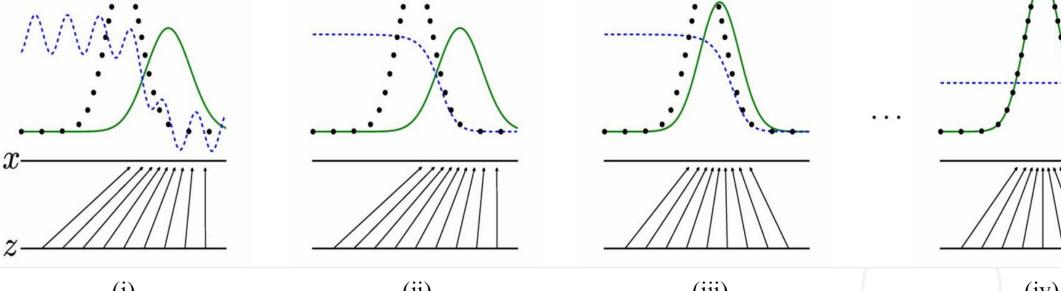


Fundamentals of GANs

- 1. First, we feed the generator with (i) noise and a (ii) training sample
- 2. We merge it and generate a new sample
- 3. The Discriminator tries to guess which of the entries is the real one
- 4. The prediction is generally in the probability interval [0,1] (e.g., sigmoid function)
- 5. The generator is rewarded if it "fools" the discriminator
- 6. The discriminator function loss is calculated (e.g., using Binary Cross Entropy (BCE))



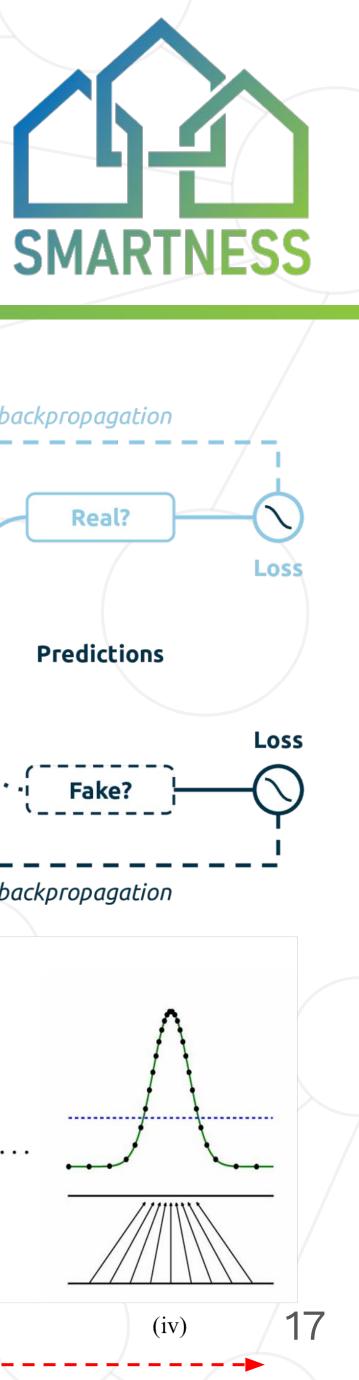


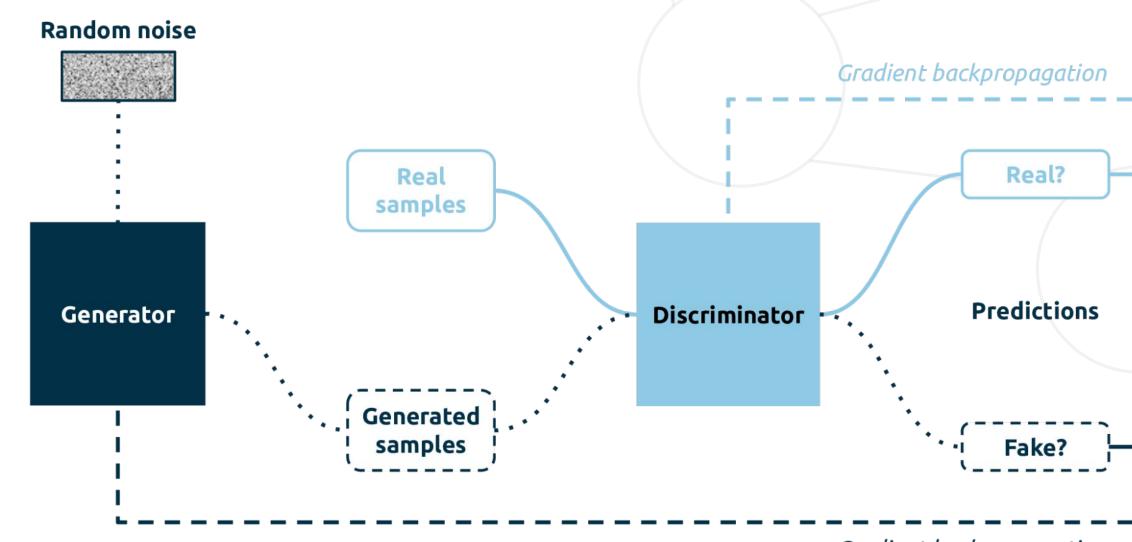


Fundamentals of GANs

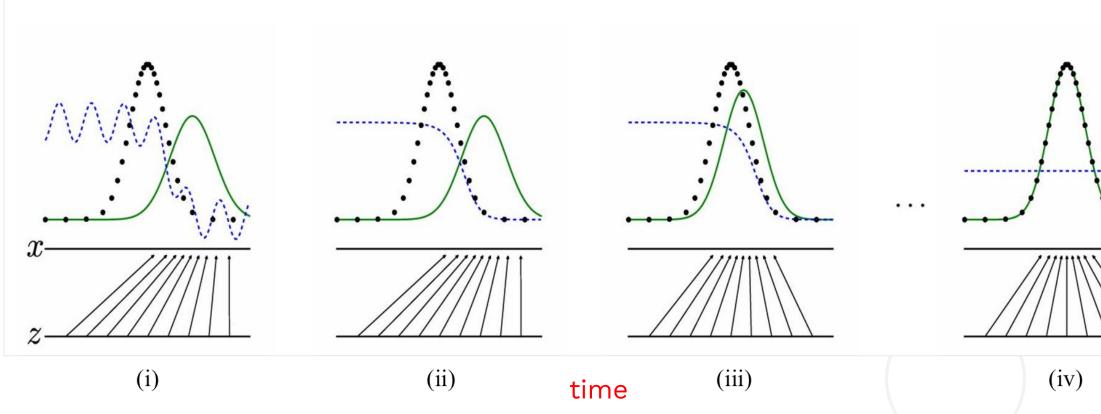
We summarize the GAN training as follows:

7. During this process, we adjust the learned distribution to the training distribution



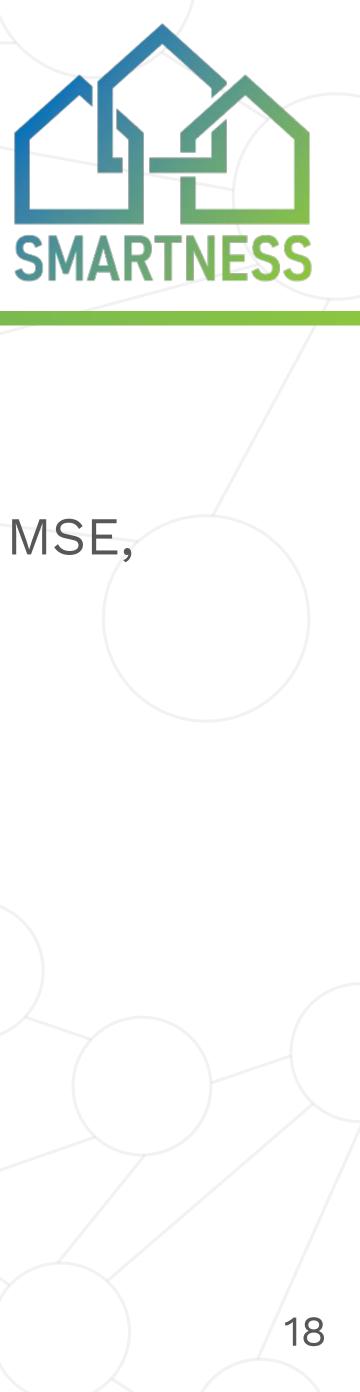


Gradient backpropagation



Challenges and limitations of GANs Fundamentals of GANs

- . Each scenario may require a different configuration of network hyperparameters:
 - Binary Cross-Entropy)

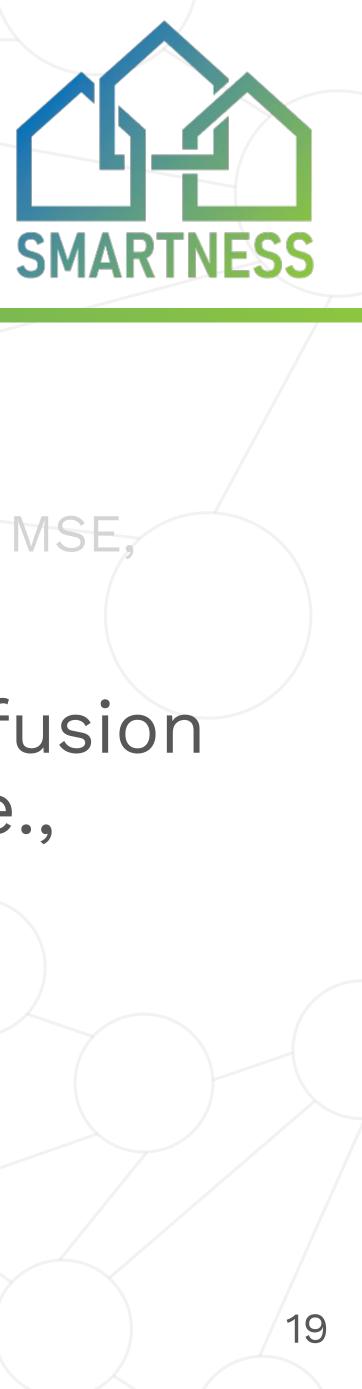


• # of neurons, optimization algorithm (e.g., Adam, SGD), Loss Function (e.g., MSE,



Challenges and limitations of GANs Fundamentals of GANs

- . Each scenario may require a different configuration of network hyperparameters: Binary Cross-Entropy)
- packet/telemetry generation? • There is not a single "silver bullet" solution



o # of neurons, optimization algorithm (e.g., Adam, SGD), Loss Function (e.g., MSE,

. Which of the existing GAN models (e.g., StyleGANs, Diffusion GANs, TimeGANs) is more suitable for our scenario - i.e.,

Challenges and limitations of GANs Fundamentals of GANs

- . Each scenario may require a different configuration of network hyperparameters:
 - Binary Cross-Entropy)
- packet/telemetry generation? o There is not a single "silver bullet" solution
- . How about the timing requirements?
 - headers)



• # of neurons, optimization algorithm (e.g., Adam, SGD), Loss Function (e.g., MSE,

. Which of the existing GAN models (e.g., StyleGANs, Diffusion GANs, TimeGANs) is more suitable for our scenario - i.e.,

• Inter- (e.g., three-way-handshake) and intra-packet (e.g., type of service in TCP

GANs with RL in Network Applications

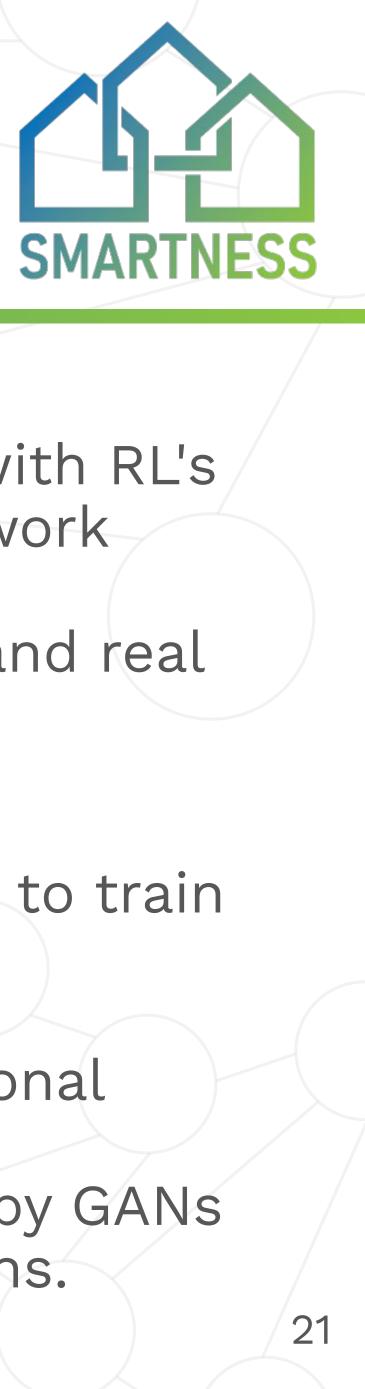
Fundamentals of GANs

Joint Applications of GANs and RL

- **Potential Benefits:** Combining GANs' ability to generate realistic data with RL's optimization capabilities offers significant potential for enhancing network configurations and policies.
- Sim-to-Real Discrepancy: Addressing differences between simulated and real network data is crucial for practical applications.

Case Studies and Research

- Automated Network Slicing: Use of GANs for generating synthetic data to train RL models, improving the efficiency of network slicing and reducing simulation-to-real discrepancies.
- **Resource Management in Network:** Integrating deep RL with distributional modeling using GANs to manage resources efficiently. to train RL algorithms for more accurate channel coefficient estimations.
- Estimating Channel Coefficients: Leveraging synthetic data generated by GANs



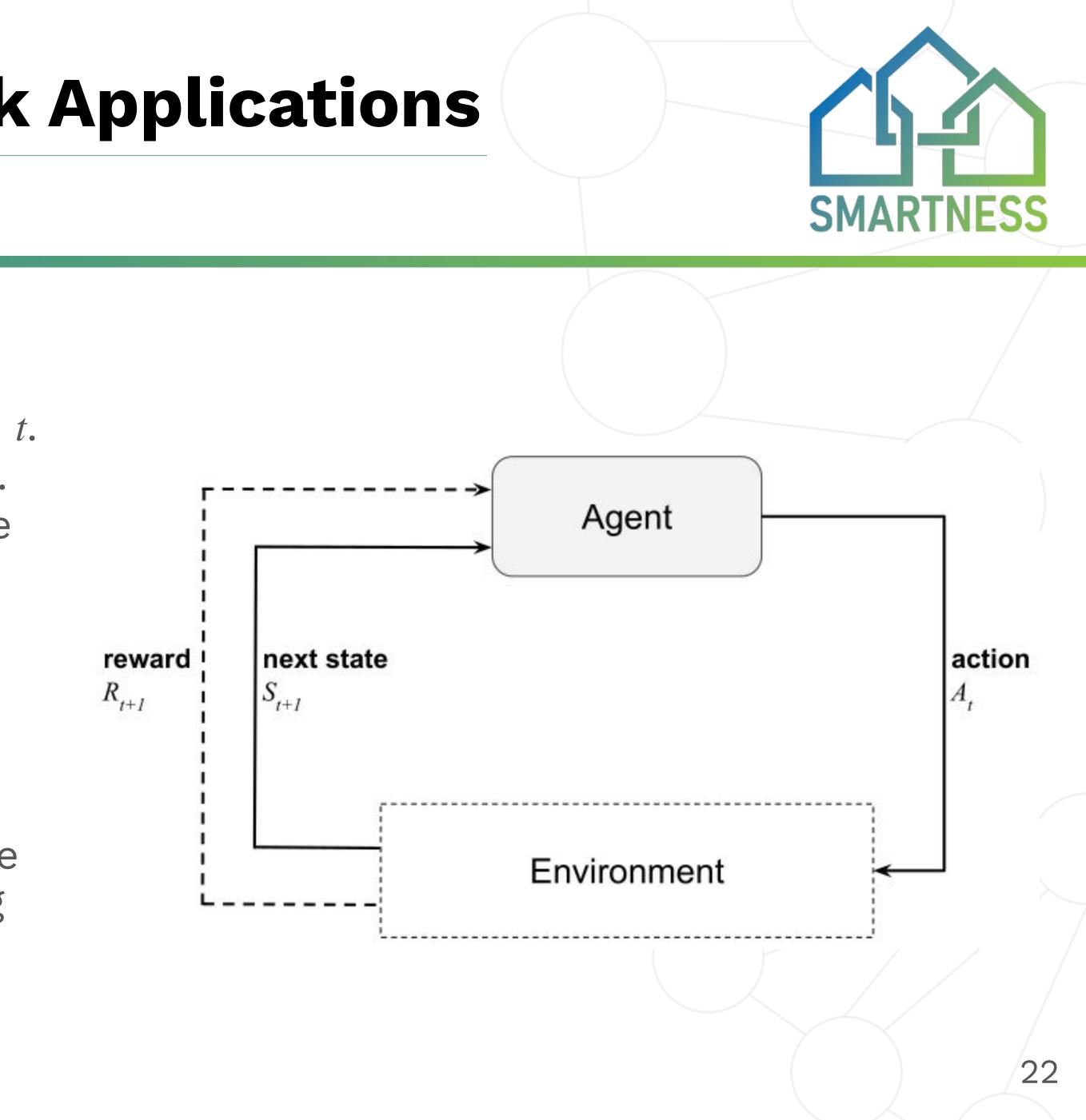
GANs with RL in Network Applications

Fundamentals of GANs

Agent-Environment Interaction in MDP (Markov decision process)

- 1. Time Step Initiation: At each specific time t.
- 2. Action Taken: The agent takes an action A.
- 3. **State Observation:** The agent observes the subsequent state S_{t+1} resulting from its action.
- 4. **Reward Assessment:** A reward value R_{t+1} is generated for each interaction, assessing the effectiveness of the action.

Objective: The process aims to maximize the reward value throughout the agent's training process, guiding the agent toward optimal decision-making.





In-band Network Telemetry and Programmable Data Plane



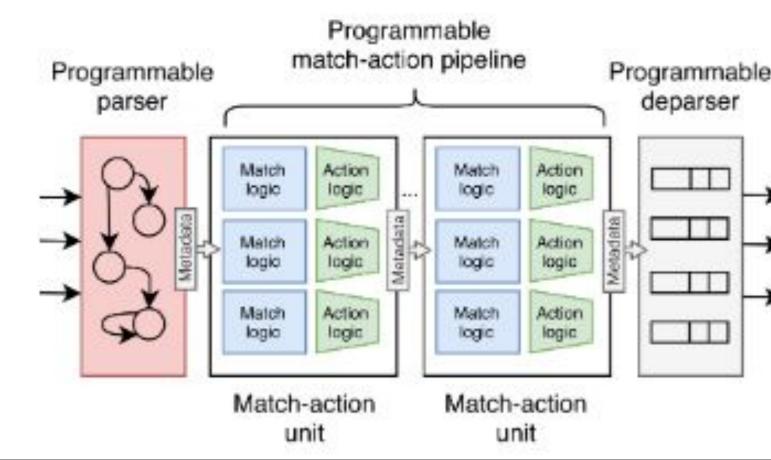


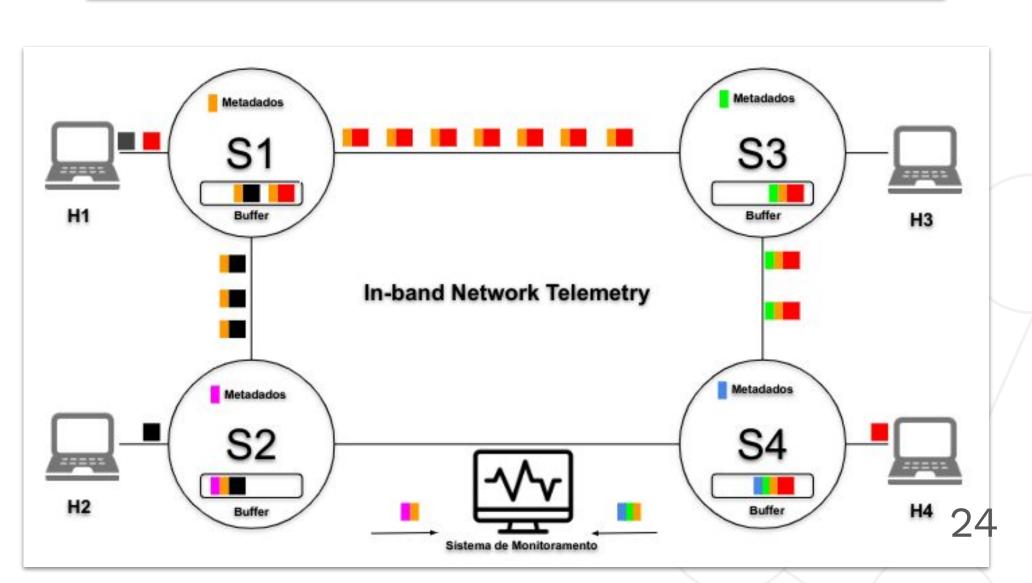
23

In-band Network Telemetry and **Programmable Data Plane**

- The data plane programmability facilitates the incorporation of intelligence during packet processing at the hardware's most proximate level, without the necessity for control plane intervention.
- Packets incorporate telemetry instructions within their header fields, facilitating the fine-grained collection and recording of network data.
- At each network hop along these paths, the data plane of the network devices employs telemetry instructions to facilitate the collection and inclusion of metadata within the packets as they traverse each node.











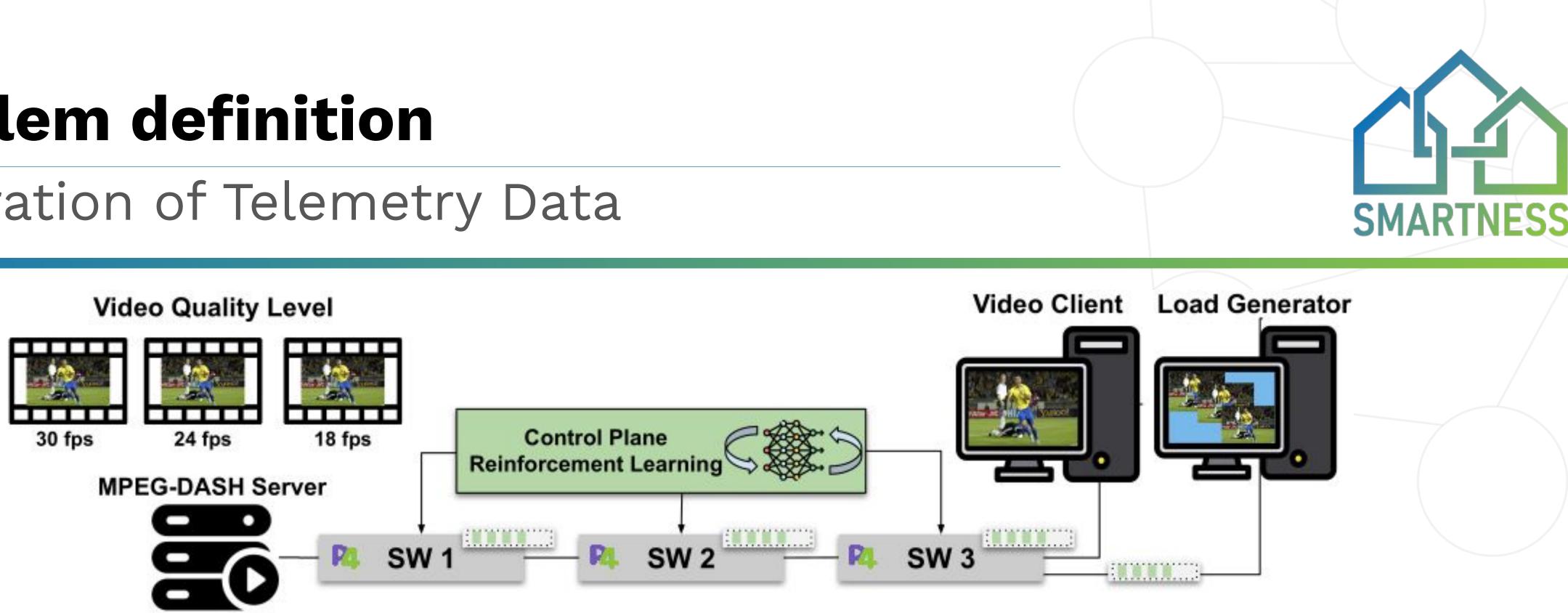
Generation of Telemetry Data



25

Problem definition

Generation of Telemetry Data



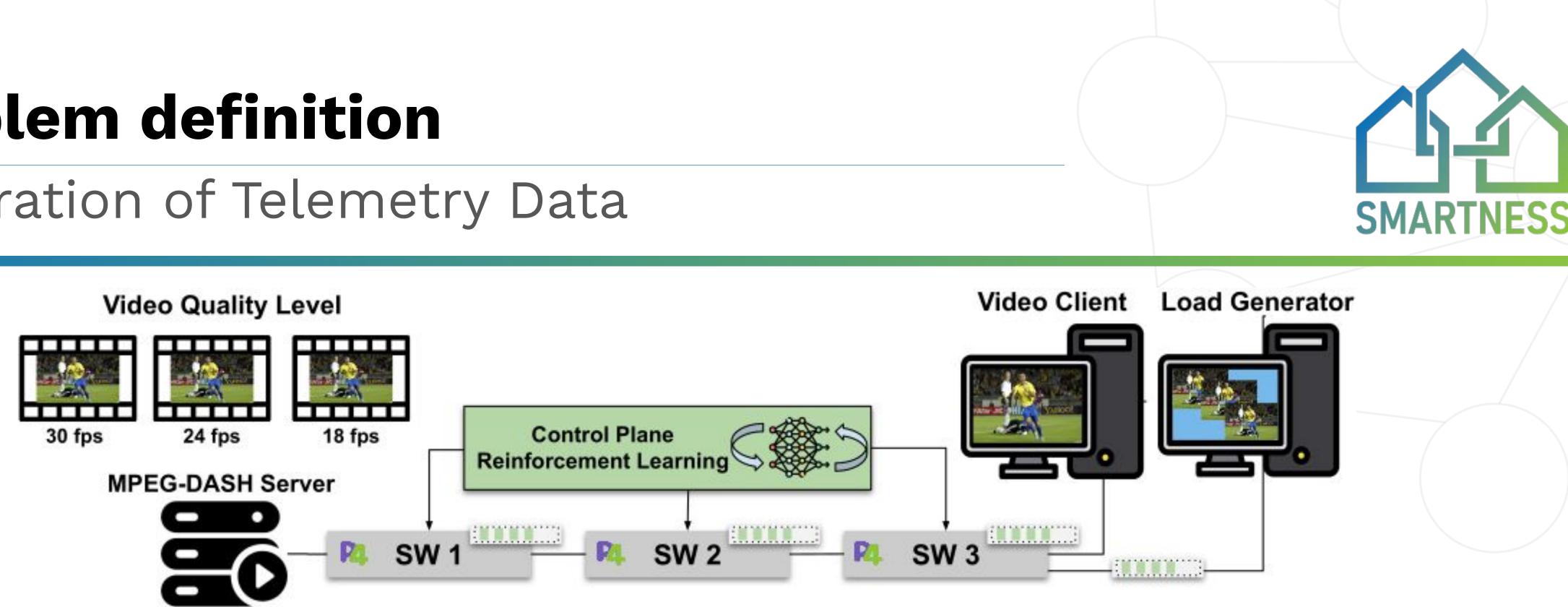
Infrastructure Overview

- Setup Components: Virtual machines interconnected via a P4 programmable data plane network.
- Application Deployment: CDN supporting MPEG-DASH for live streaming a soccer game.
- Load Management: WAVE, is versatile load generator used for orchestrating application instances over time.
- Network Architecture: Includes three programmable switches collecting INT telemetry data, complemented by a Video Client for metrics.



Problem definition

Generation of Telemetry Data



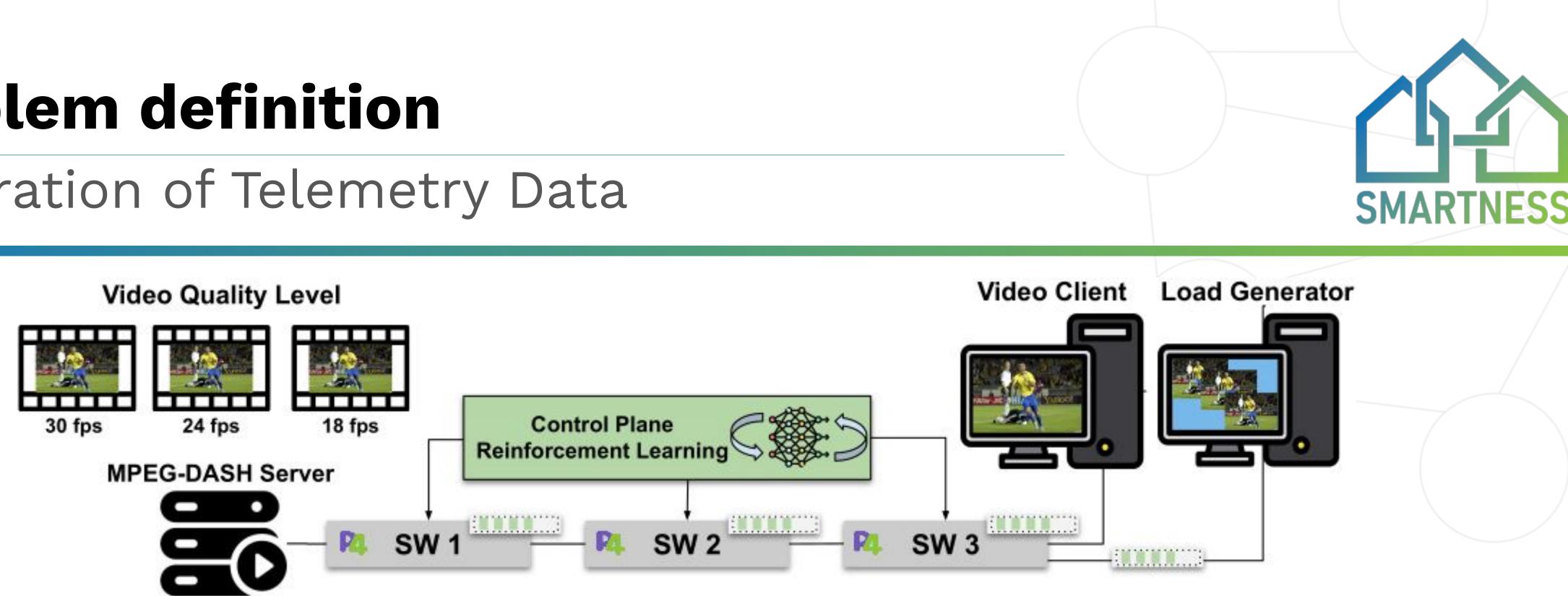
Role of RL Agent

- Primary Function: Operates as a data plane optimizer, managing queue sizes in switches to enhance user experience.
- Goal: Optimize resource utilization to improve network infrastructure efficiency.



Problem definition

Generation of Telemetry Data

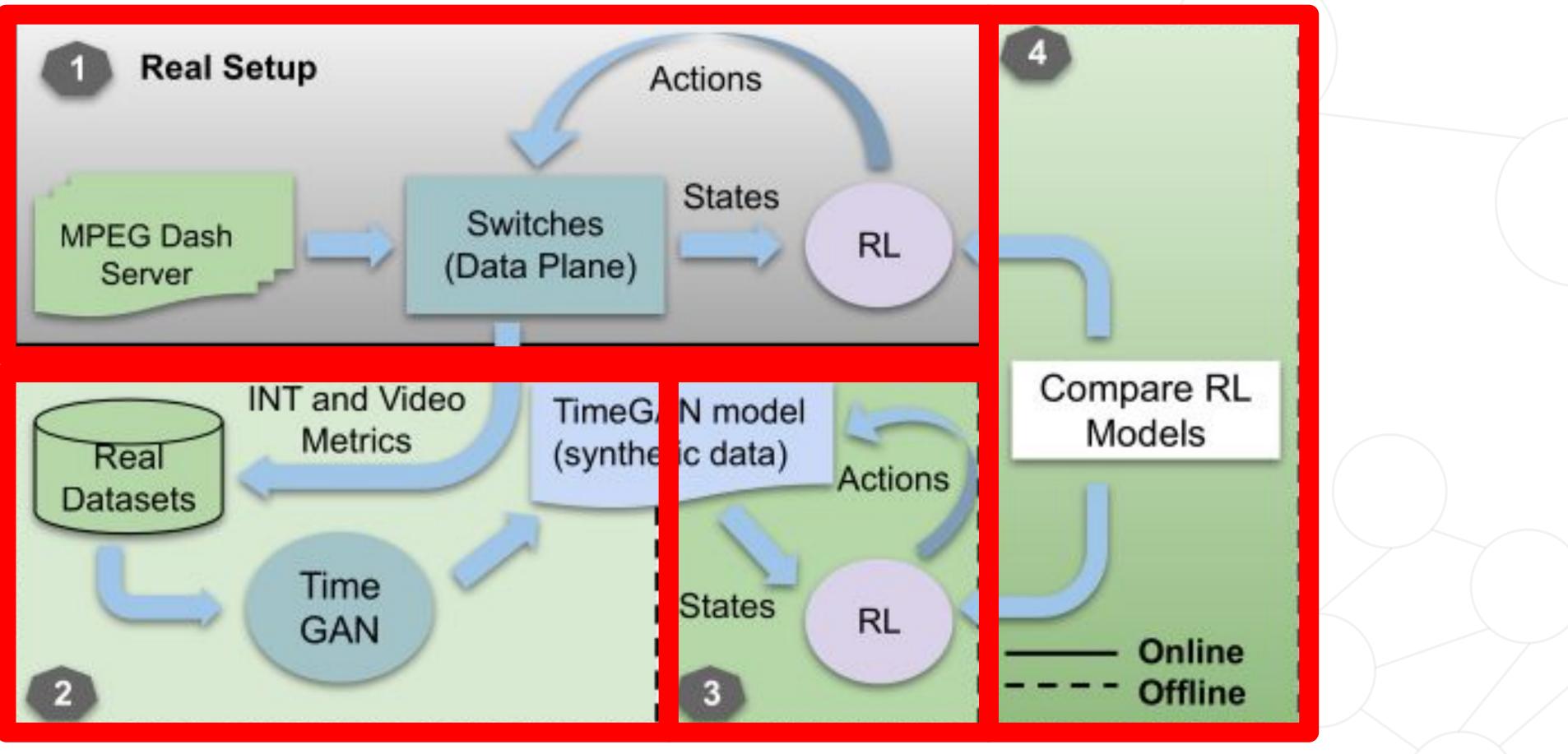


Data Plane Optimization

- Challenges: Gaining cooperation from network operators for experiments can be difficult.
- Alternative Approach: Use a GAN trained on real data as a simulator to train the RL agent without needing a real setup.



Methodology Generation of Telemetry Data





Methodology

Generation of Telemetry Data

Training the RL Model

- network.
- Synthetic Scenario: Offline training of the RL model using synthetic data from the TimeGAN to assess generalization capacity.

Importance of GAN in RL Training

- Dataset Issues: Original datasets may have imbalances, inadequacies, or erroneous values.
- GAN Advantages: Provides the ability to generate balanced, comprehensive data for diverse experimental scenarios.
- training using synthetic versus real data.



• **Real Setup:** RL agent trained using real data collected from the

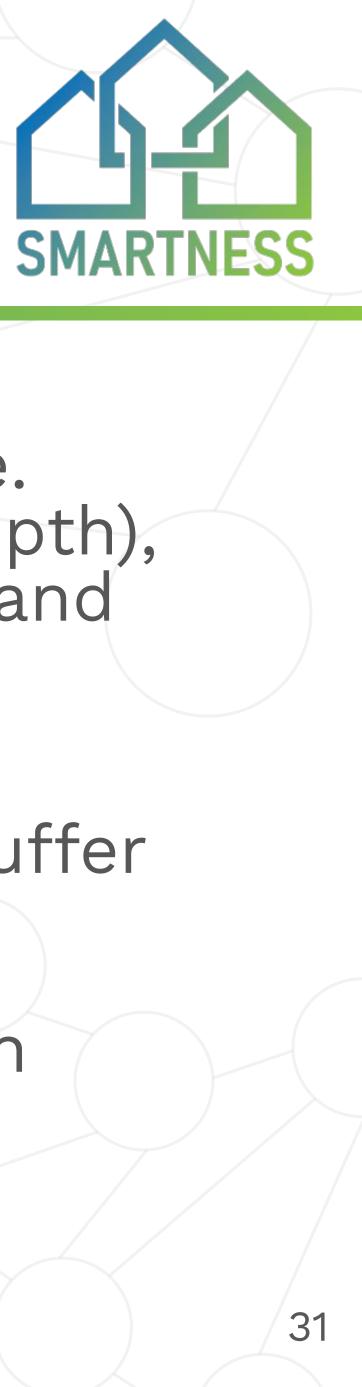
• Efficiency Comparison: Evaluates the time efficiency of RL model

Dataset Characterization

Generation of Telemetry Data

Dataset Composition

- Video Metrics: Frames per second (FPS), bitrate, buffer size.
- **Experiments and Data Collection**
- Buffer Size Configurations: Two experiments with switch buffer sizes set at 32 and 64 packets.
- Data Merging and Filtering: Datasets merged based on timestamps, filtering out higher `Deq Timedelta' to focus on high-load conditions.



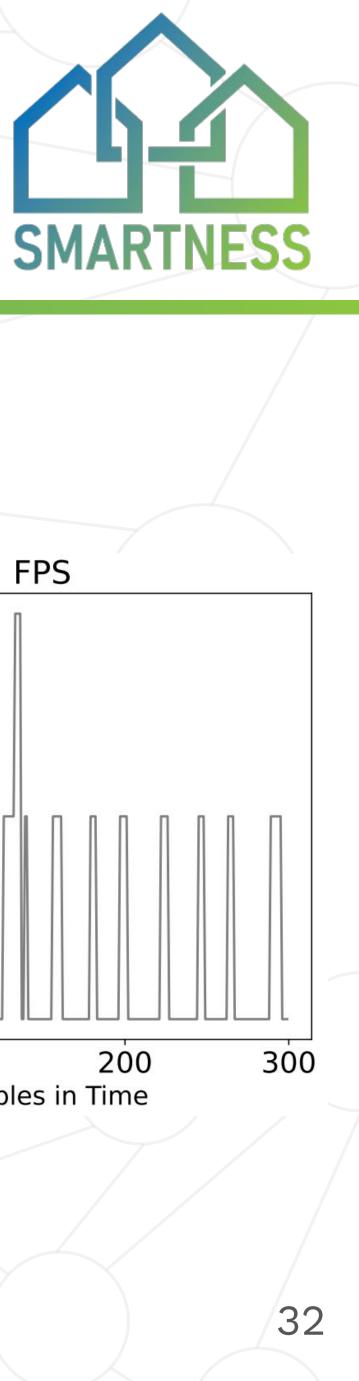
• **Network Metrics:** Queue depth at packet queuing (Enq Qdepth), packet queuing duration in microseconds (Deq Timedelta), and queue depth at packet dequeuing (Deq Qdepth).

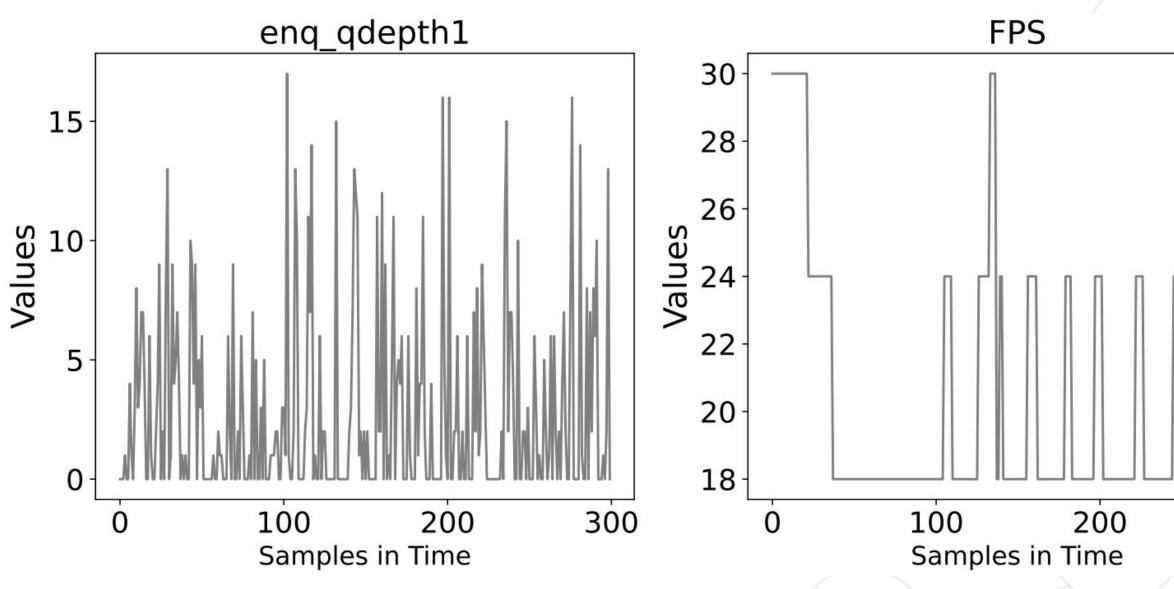
Dataset Characterization

Generation of Telemetry Data

Challenge of Non-Stationary Data

- Non-Stationarity: features within the Programmable Data Plane application exhibit non-stationary characteristics.
- Visualization: Non-stationarity visually demonstrated in the figure alongside.
- Implication: This nature complicates direct comparison between real telemetry data and synthetic data generated by GANs.





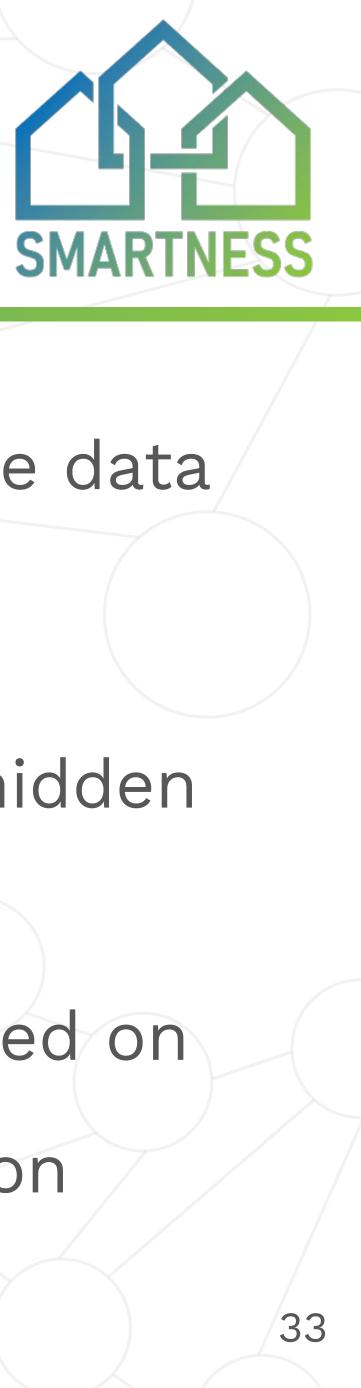
TimeGAN Training

Generation of Telemetry Data

Data Preprocessing

Hyperparameter Configuration

- *Importance:* Critical for optimizing the training regimen. *Parameters:* Sequence sizes, sequence length, number of hidden dimensions, batch size.
- Hyperparameter Tuning
- Method: Empirical approach with iterative adjustments based on training outcomes and insights. • Challenges: Identifying optimal settings due to the impact on
- model performance.



• **Steps:** Address missing data, remove outliers, and normalize data to prepare for effective training.

Model Selection

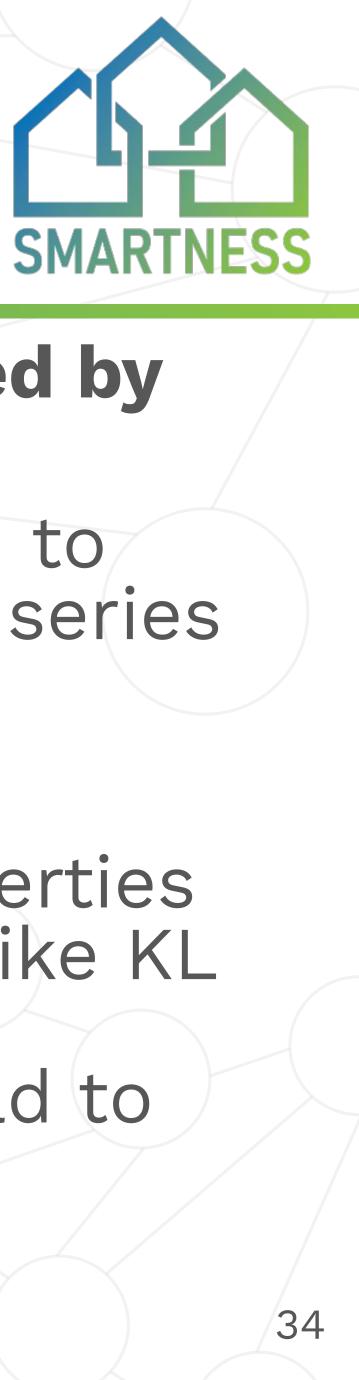
Generation of Telemetry Data

Lack of Consensus in Evaluating Synthetic Data Generated by GANS

• Highlighted in (Brophy, 2023), there is no agreed method to data.

Complexity with Non-Stationary Data

- divergence.
- misleading results when using traditional metrics.



assess distributions created by GANs, specially for time series

• Non-stationary time series show varying statistical properties over time, complicating traditional evaluation methods like KL

• Real data variability vs. synthetic data constancy can lead to

Model Selection

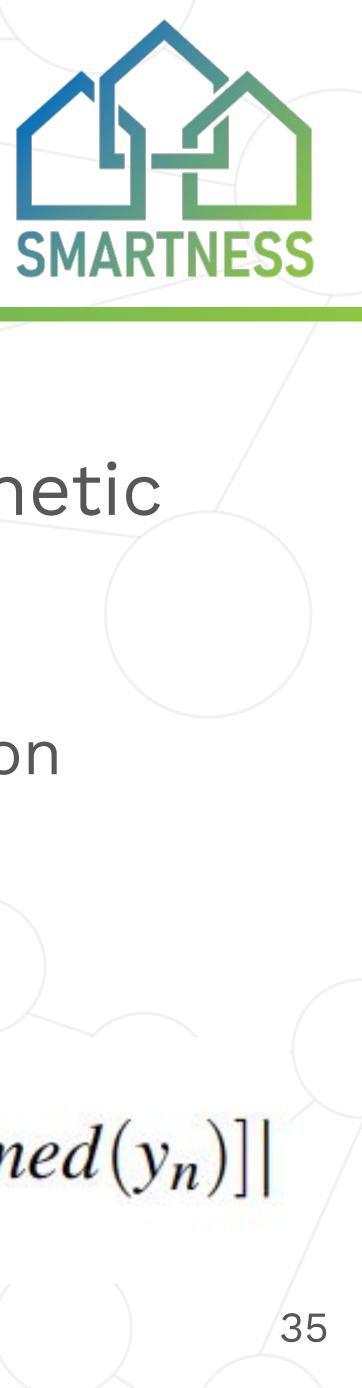
Generation of Telemetry Data

Developing a New Metric

data distributions, focusing on statistical measures.

- Metric Calculation:
 - between real (X) and synthetic (y) datasets.
 - Median Difference: Addresses positional differences between distributions.

n_feats $\mathbf{M} = \sum |[Q_3(X_n) - Q_1(X_n)] - [Q_3(y_n) - Q_1(y_n)]| + |[med(X_n) - med(y_n)]|$ n=1



Designed to assess the similarity between real and synthetic

• Interguartile Discrepancy: Measures the difference in dispersion

Hands on Two Possibilities







36

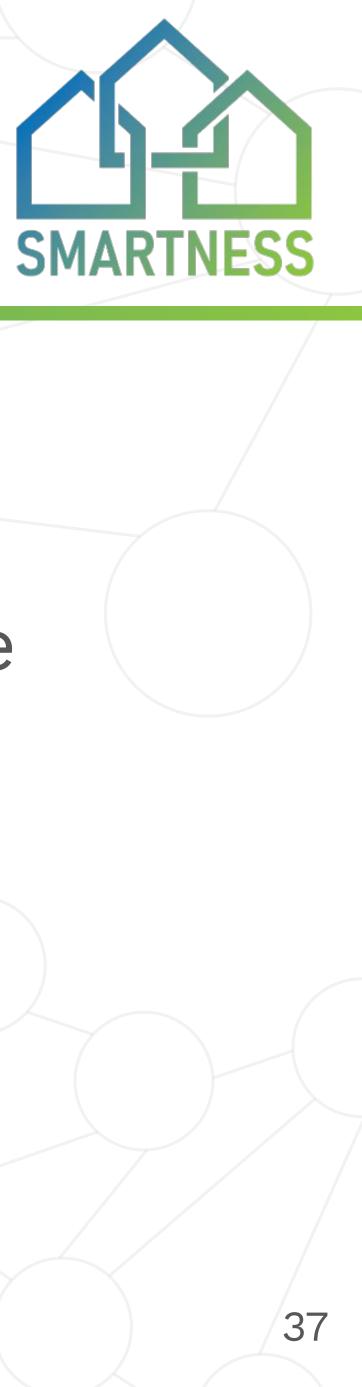
GitHub

Generation of Telemetry Data

- Clone the following project from GitHub: o https://github.com/thiagocaproni/tutorial-timegan
- After cloning the project from GitHub, create the environment by running the following command (where 'environment.yml' is located)

o conda env create

• Then, type the following command: o conda activate ydata

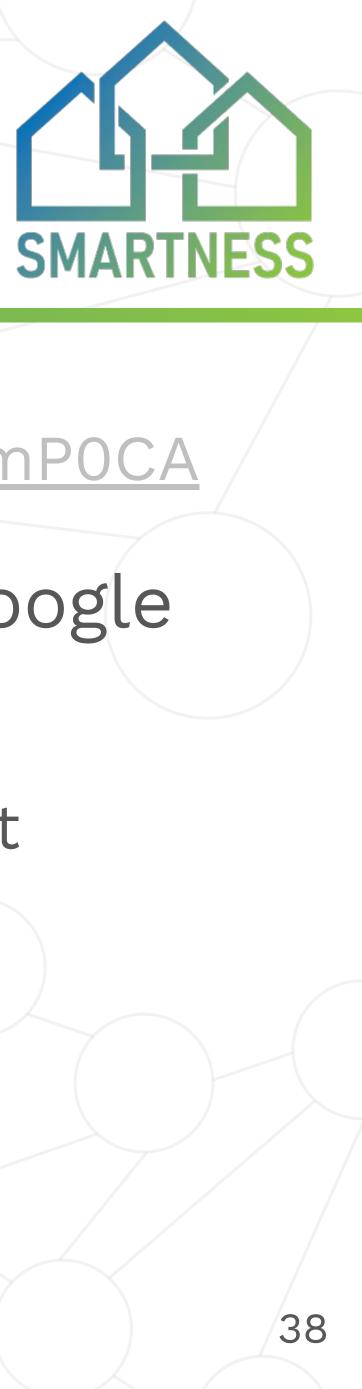


Google Colab

Generation of Telemetry Data

1. Download the zip file from the following link: o https://drive.google.com/file/d/1kR0teCHU4Z2jCez75kcM61GArmP0CA

- fY/view?usp=sharing
- Drive root folder
 - All paths used in Python scripts and notebooks are executed directory.
- 3. Navigate to the following folder
 - tutorial -> code -> timegan
- 4. Open the notebook:
 - o main.ipynb



2. Unzip the file and upload the "tutorial" folder to the Google

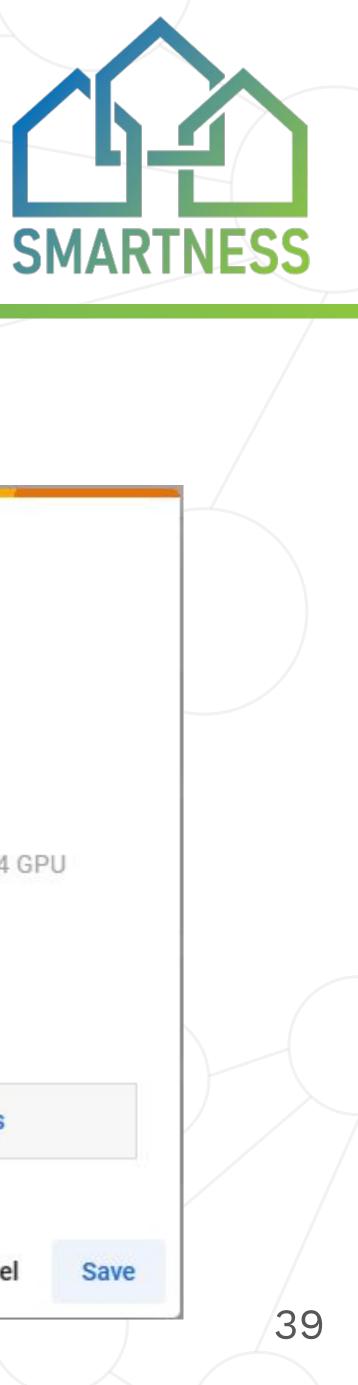
considering that the "tutorial" folder is in the Google Drive root

Google Colab

Generation of Telemetry Data

5. Make sure, you are using a Python 3 session:

File Edit View Insert	Runtime Tools Help Last	saved at 2:30 PM
+ Code + Text	Run all	Ctrl+F9
	Run before	Ctrl+F8
 TimeGAN G 	Run selection	Ctrl+Shift+Enter
	Run after	Ctrl+F10
This notebook serves a steps, including prepro	Interrupt execution	Ctrl+M
	Restart session	Ctrl+M
	Restart session and run all	1
 Environment 	Disconnect and delete runtim	e
[] # Install Ydata	Change runtime type	
pip install yda	Manage sessions	
	View resources	
[] # Mount Google D from google.cola	view runtime logs	



Ру	thon 3		•					
Hardware	e accelera	tor (2					
0	CPU	0	T4 GPU	0	A100 GPU	0	L4 GPU	
0	V100 G	PU (de	precated)	0	TPU (depre	cated)		
0	TPU v2							
Want ac	cess to p	remiun	n GPUs? Pu	urchase	additional co	mpute u	nits	

Google Colab

Generation of Telemetry Data

6. The first part is + Código + Texto compound with cells TimeGA V to setup the This notebook se steps, including p environment:



[] 4 2 células ocu

Generate >

[] 4 1 célula ocul

> Evaluate t

[] 4 1 célula ocult



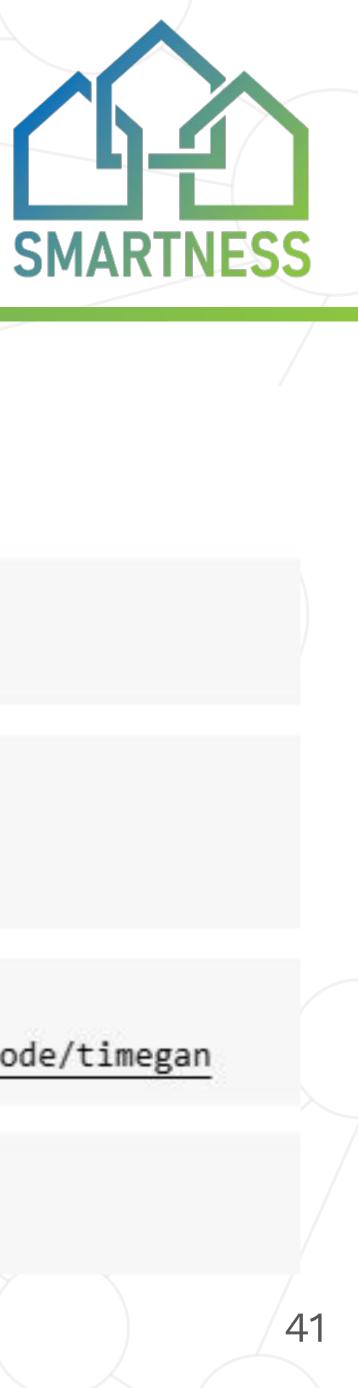
	SMARTNESS
N Generation erves as the primary workflow file for the TimeGAN synth preprocessing, training, data generation, and evaluation of	netic data generation process. It enables us to execute all necessary of the synthetic datasets.
nent Setup	
rultas	
the models	
rultas	
the synthetic data	
ılta	
the generated models	
ilta	40
	40

Open main notebook

Generation of Telemetry Data

7. Run the cells to setup the environment in order to:

- a. Install the ydata-synthetic module (probably you be asked to restart the session)
- b. Mount the Google Drive folder
- c. Access the "timegan" folder
- d. Append the "data_process" folder where is located the script to preprocessing



Environment Setup

[] # Install Ydata Synthetic !pip install ydata-synthetic==1.1.0

[] # Mount Google Drive folder from google.colab import drive drive.mount('/content/gdrive')

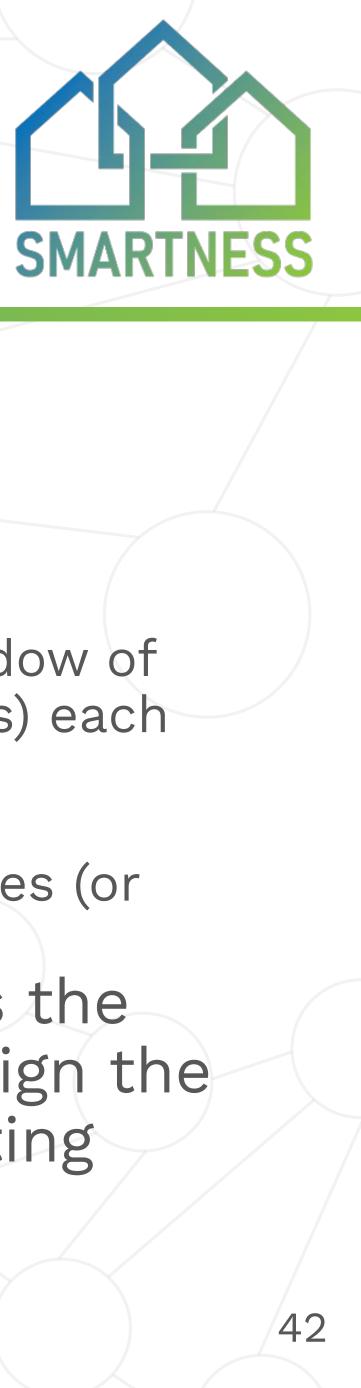
[] # Access the folder timegan %cd /content/drive/MyDrive/tutorial/code/timegan

[] import sys
 sys.path.append('../data_process/')

Parameters file

Generation of Telemetry Data

- 1. Open the file: (code/timegan/params.py)
 - a. The script has the amount of models variable
 - b. Its value is factored to define the values of the following hyperparameters:
 - sequence contains.
 - hidden dim (j): Number of units or neurons in each hidden layer
 - models.



seq len (i): the sequence length would be the size of the temporal window of each sequence used to train the model, that is, how many time steps (lines) each

batch size (k): The batch size determines how many temporal sequences (or how many data examples/lines) are included in a single batch for training. c. The fatNum function in the script model utility.py returns the values of i, j and k that is used in several other scripts to assign the hyperparameters variations for training, generating and evaluating

Preprocessing (code/data_process/preprocess_data.py) Generation of Telemetry Data

def loadDataSet(self, path_int, path_dash):

Load the INT and DASH datasets from specified paths df_int = pd.read_csv(path_int, sep = ',') df_dash = pd.read_csv(path_dash, sep = ';')

from milliseconds to seconds self.transformTimeStamp(df_dash) df_int = df_int.loc[df_int.groupby('timestamp')['deq_timedelta1'].idxmax()]

Set 'timestamp' as the index for the INT dataset df_int.set_index('timestamp', inplace=True)

Merge the INT and DASH datasets on their timestamp indices and reset the merged DataFrame's index self.dataset = pd.merge(df_int, df_dash, left_index=True, right_index=True).reset_index()





Preprocessing (code/data_process/preprocess_data.py) Generation of Telemetry Data

```
def preProcessData(self, num_cols, cat_cols, random):
   # Fill missing values in numerical columns with their mean
   for i in num_cols:
       self.dataset[i].fillna(self.dataset[i].mean(), inplace=True)
   # Perform one-hot encoding if there are categorical columns
   if len(cat_cols) > 0:
       self.hotEncode()
       cat_cols = [0,1,2]
   # Create a copy of the dataset with only the processed columns
   self.processed_data = self.dataset[ num_cols + cat_cols ].copy()
   self.cat_cols = cat_cols
   self.num_cols = num_cols
    # Randomly shuffle the dataset if requested
   if random == True:
       idx = np.random.permutation(self.processed_data.index)
       self.processed_data = self.processed_data.reindex(idx)
```



```
def loadDp(random, outliers):
   dp = DataPre()
   #Loading and mergint INT and DASH datasets
   dp.loadDataSet(path_int='../../datasets/log_INT_TD-32_100.csv',
                  path dash='../../datasets/dash TD-32 100.csv')
    #preprocessing data
   dp.preProcessData(params.num_cols, cat_cols=params.cat_cols, random=random)
    #removing columns with same values
    dp.removeSameValueAttributes()
   if outliers == False:
       dp.removeOutliers()
    #printing processed data
   dp.processed_data
```

return dp





def train(dp, seq_len, n_seq, hidden_dim, noise_dim, dim, batch_size, model, train_steps): learning_rate = 5e-4

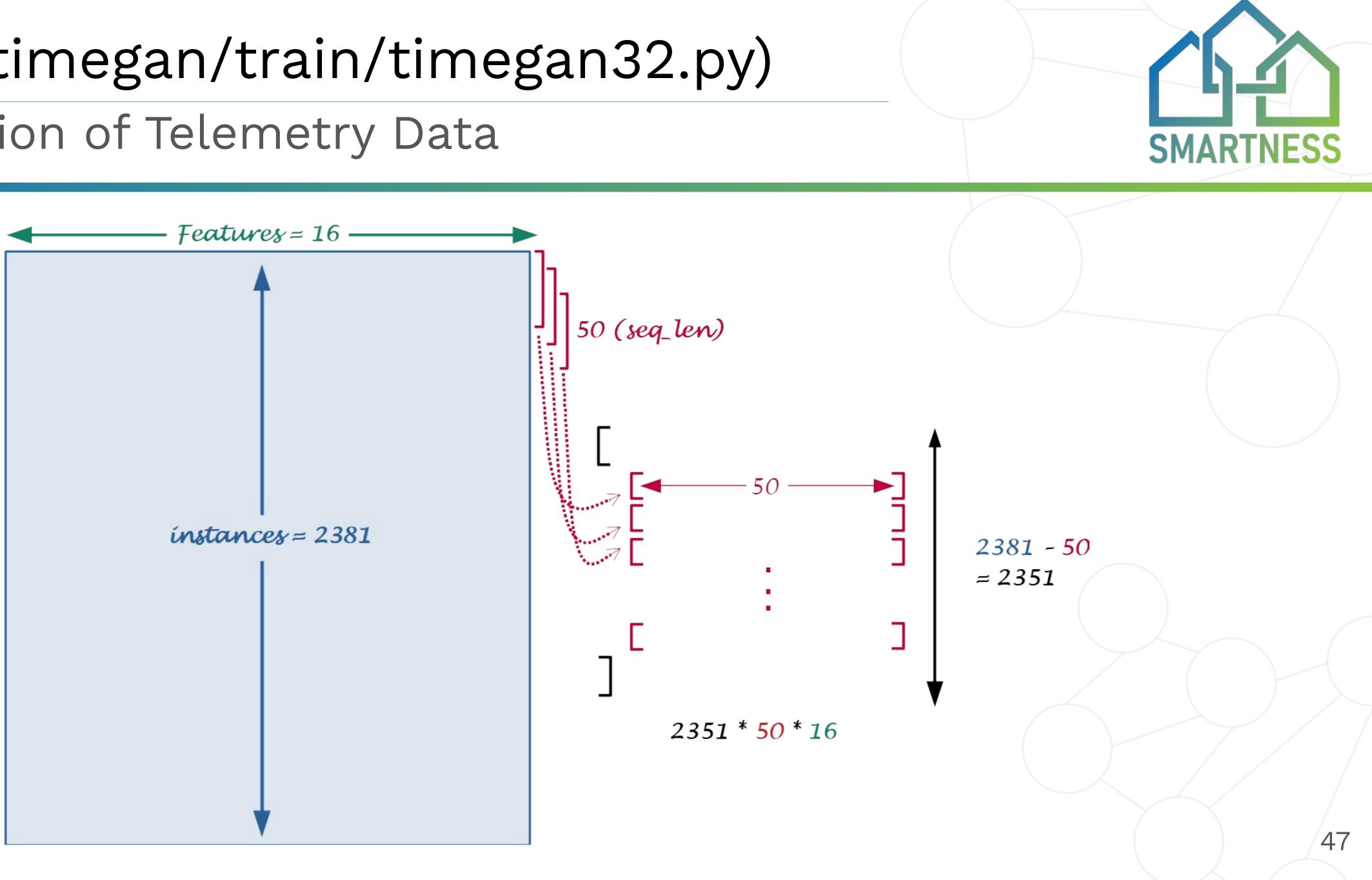
gan_args = ModelParameters(batch_size=batch_size, lr=learning_rate, noise_dim=noise_dim, layers_dim=dim)

#normalizing the data processed_data = real_data_loading(dp.processed_data.values, seq_len=seq_len)

synth.train(processed_data, train_steps=train_steps) synth.save(model)



synth = TimeGAN(model_parameters=gan_args, hidden_dim=hidden_dim, seq_len=seq_len, n_seq=n_seq, gamma=1)





```
dp = loadDp(random=False, outliers=False)
iMax, jMax, kMax = ModelUtility.fatNum(params.amount_of_models) # Change the file params.py
print("\nNumber of models" + str(params.amount_of_models) + ' iMax: ' + str(iMax) + ' jMax: ' + str(jMax) + ' kMax: ' + str(kMax))
print("Num GPUs Available: ", len(tf.config.list_physical_devices('GPU')))
try:
  # Specify an valid GPU device
  with tf.device('/device:GPU:0'):
    for i in range(0,iMax):
      for j in range(0,jMax):
        for k in range(0,kMax):
          train(dp,
            seq_len=(50*(i)+50),
            n_seq=params.merged_columns_len,
            hidden_dim=(20*(j)+20),
            noise_dim=32,
            dim=128,
            batch_size=(28*(k) + 100),
            model=str('../saved_models/so32_seqlen_'+ str((50*(i) + 50)) + '_hidim_' + str(20*(j)+20) + '_batch_' + str(28*(k) + 100) + '.pkl'),
            train_steps=params.train_steps)
except RuntimeError as e:
  print(e)
```



def loadSynthData(model32, model64, number_of_windows): synth_32 = TimeGAN.load(model32)

synth_data_32 = synth_32.sample(number_of_windows)

synth_64 = TimeGAN.load(model64) synth_data_64 = synth_64.sample(number_of_windows)

synth_data_32[:,:,13:16][synth_data_32[:,:,13:16] >= 0.5] = 1 synth_data_32[:,:,13:16][synth_data_32[:,:,13:16] < 0.5] = 0</pre> synth_data_64[:,:,13:16][synth_data_64[:,:,13:16] >= 0.5] = 1 synth_data_64[:,:,13:16][synth_data_64[:,:,13:16] < 0.5] = 0</pre>

return synth_data_32, synth_data_64





```
def loadRealData(dsint32, dsint64, dsdash32, dsdash64, num_cols, cat_cols, sample_size, randon, outliers):
   dp32 = DataPre()
   dp32.loadDataSet(path_int=dsint32, path_dash=dsdash32)
   dp32.preProcessData(num_cols, cat_cols=cat_cols, random=randon)
   if outliers == False:
       dp32.removeOutliers()
   real_data_32 = dp32.processed_data
    real_data_32 = real_data_32[0:sample_size].copy()
   real_data_32 = real_data_32.values
    #loading 64 bit buffer dataset
   dp64 = DataPre()
   dp64.loadDataSet(path_int=dsint64, path_dash=dsdash64)
   dp64.preProcessData(num_cols, cat_cols=cat_cols, random=randon)
   if outliers == False:
       dp64.removeOutliers()
    real_data_64 = dp64.processed_data
    real_data_64 = real_data_64[0:sample_size].copy()
    real_data_64 = real_data_64.values
```

```
return real_data_32, real_data_64
```



```
def getStatistics(data):
   median = np.median(data)
    percentile_25 = np.percentile(data, 25)
    percentile_75 = np.percentile(data, 75)
   return [percentile_25, median, percentile_75]
```

```
def genStatisctics(real_32, synth_32, real_64, synth_64, sample_size, num_cols):
   dict = \{\}
```

```
for j, col in enumerate(num_cols):
   dict[col] = [getStatistics(real_32[:,j][:sample_size]),
                 getStatistics(synth_32[:,j][:sample_size]),
                 getStatistics(real_64[:,j][:sample_size]),
                 getStatistics(synth_64[:,j][:sample_size])]
```



```
def createDataSet(seq_len, data):
    lines = int(params.synth_sample_size/seq_len)
```

```
for i in range(0,lines):
    for j in range(0, seq_len):
        dataset[(i*seq_len) + j] = data[i][j][:]
```

return dataset



dataset = np.zeros(lines * seq_len * params.merged_columns_len).reshape(lines*seq_len, params.merged_columns_len)

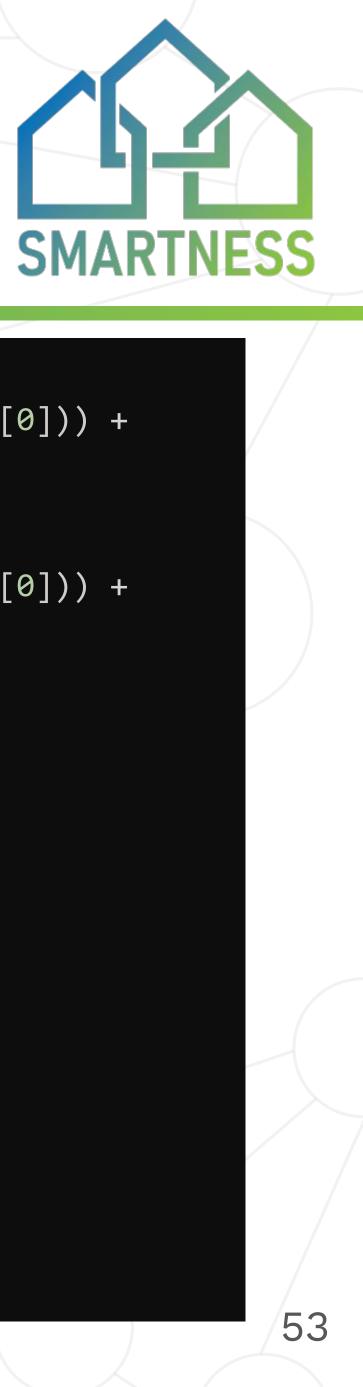
def getMetrics(statistic_data):

abs(statistic_data[0][1] - statistic_data[1][1]))

abs(statistic_data[3][1] - statistic_data[2][1]))

return metric32, metric64

def get_allfeatures_metrics(metrics, model_index, statistic_data): for j, col in enumerate(params.num_cols): metrics[0][model_index][j], metrics[1][model_index][j] = getMetrics(statistic_data.get(col))



- metric32 = (abs((statistic_data[0][2] statistic_data[0][0]) (statistic_data[1][2] statistic_data[1][0])) +
- metric64 = (abs((statistic_data[2][2] statistic_data[2][0]) (statistic_data[3][2] statistic_data[3][0])) +

Model Selection code/timegan/evaluation/analyze_data_models.ipynb Generation of Telemetry Data

```
def getFeaturesBestMetricsOfModels(models, metrics):
   sum32, sum64 = sumFeatureMetricsOfModels(models, metrics)
   index = np.argmin(sum32)
   model = getModelNameByIndex(index)
   #print('bestmodel_int32: ' + model + ' index: ' + str(index))
   best_32 = models.get(model)[0]
   index = np.argmin(sum64)
   model = getModelNameByIndex(index)
   #print('bestmodel_int64: ' + model + ' index: ' + str(index))
   best_64 = models.get(model)[0]
   index = np.argmax(sum32)
   model = getModelNameByIndex(index)
   #print('worst_int32: ' + model + ' index: ' + str(index))
   worst_32 = models.get(model)[0]
   index = np.argmax(sum64)
   model = getModelNameByIndex(index)
   #print('worst_int32: ' + model + ' index: ' + str(index))
   worst_64 = models.get(model)[0]
```

return best_32, worst_32, best_64, worst_64



Model Selection code/timegan/evaluation/analyze_data_models.ipynb Generation of Telemetry Data

```
def sumFeatureMetricsOfModels(models, data_metrics):
    sum32 = np.zeros(len(models))
    sum64 = np.zeros(len(models))
    for i in range(len(models)):
        sum32[i] = sum(data_metrics[0,i,:])
        sum64[i] = sum(data_metrics[1,i,:])
    return sum32, sum64
```





Applying synthetic data to an RL agent

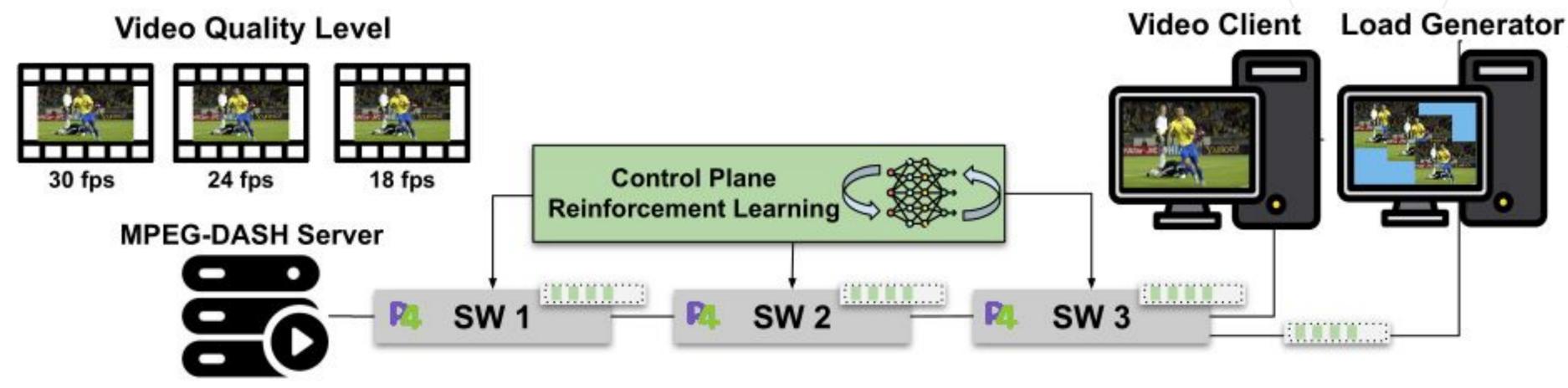






Applying synthetic data to an RL agent Generation of Telemetry Data

Real setup



Challenges

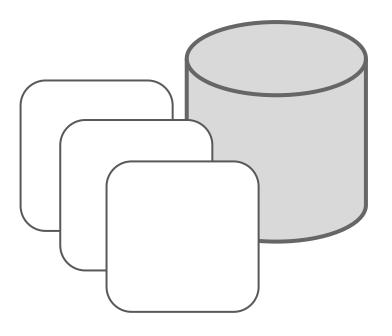
- The agent training time depends on the video streaming duration.



• The infrastructure requirements may not be feasible for a real setup;

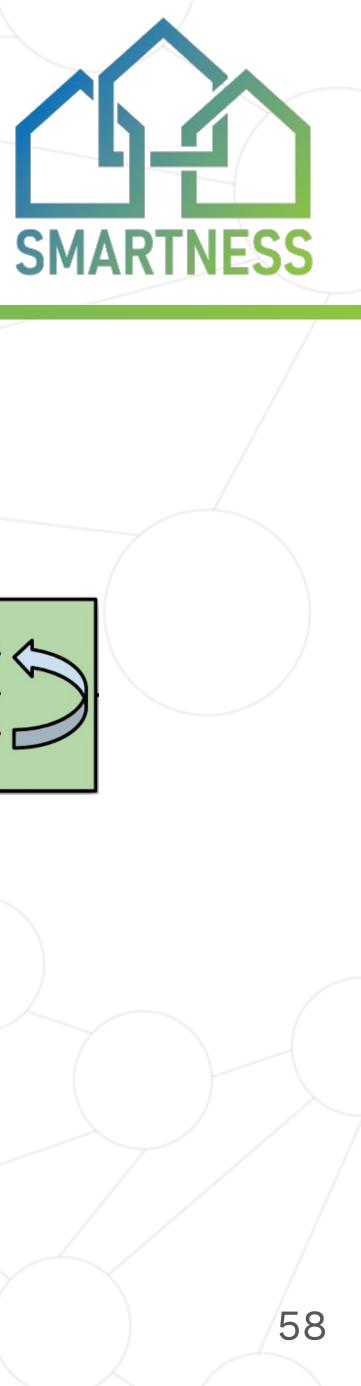
Applying synthetic data to an RL agent Generation of Telemetry Data

Possible solution



Synthetic data

INT metadata and QoS metrics

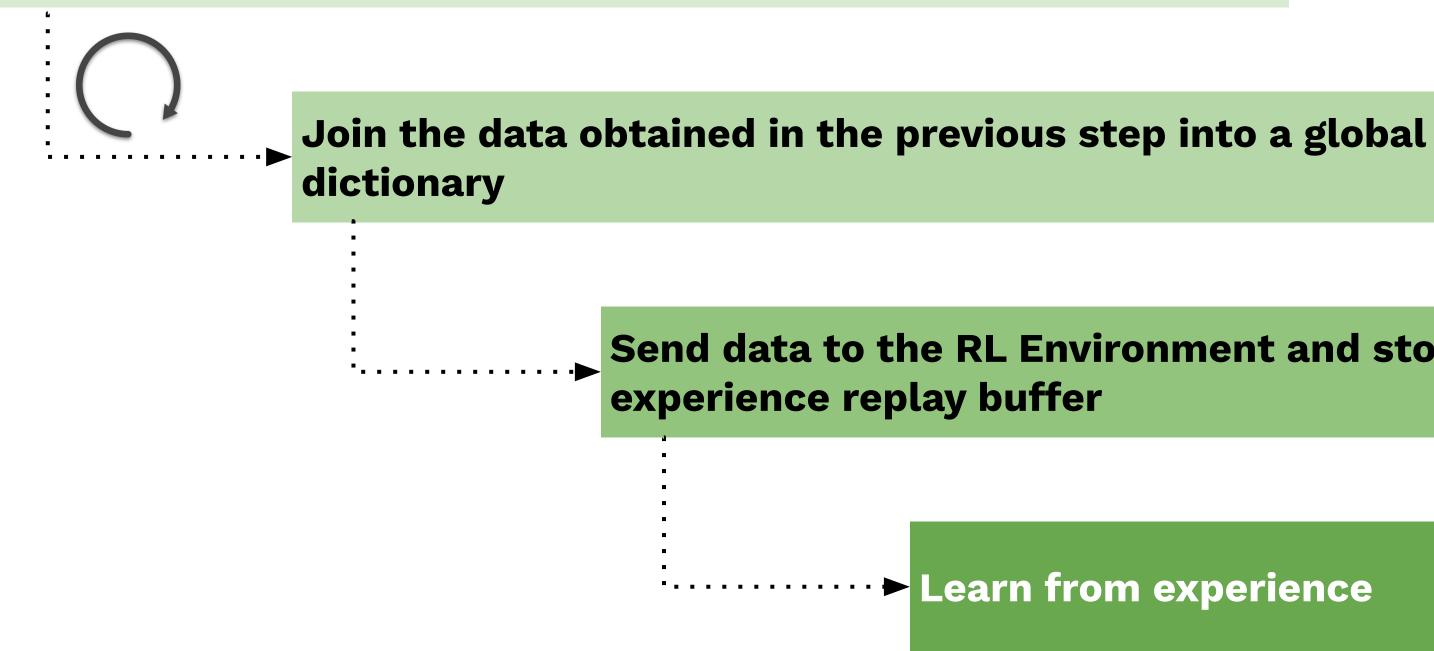




Applying synthetic data to an RL agent Generation of Telemetry Data

How can we implement it?

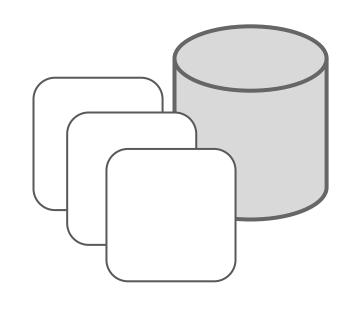
Read INT metadata regarding the 32-bit and 64-bit queue sizes from their respective CSV files





Send data to the RL Environment and store the transition in an

Applying synthetic data to an RL agent Simulating the real network behavior



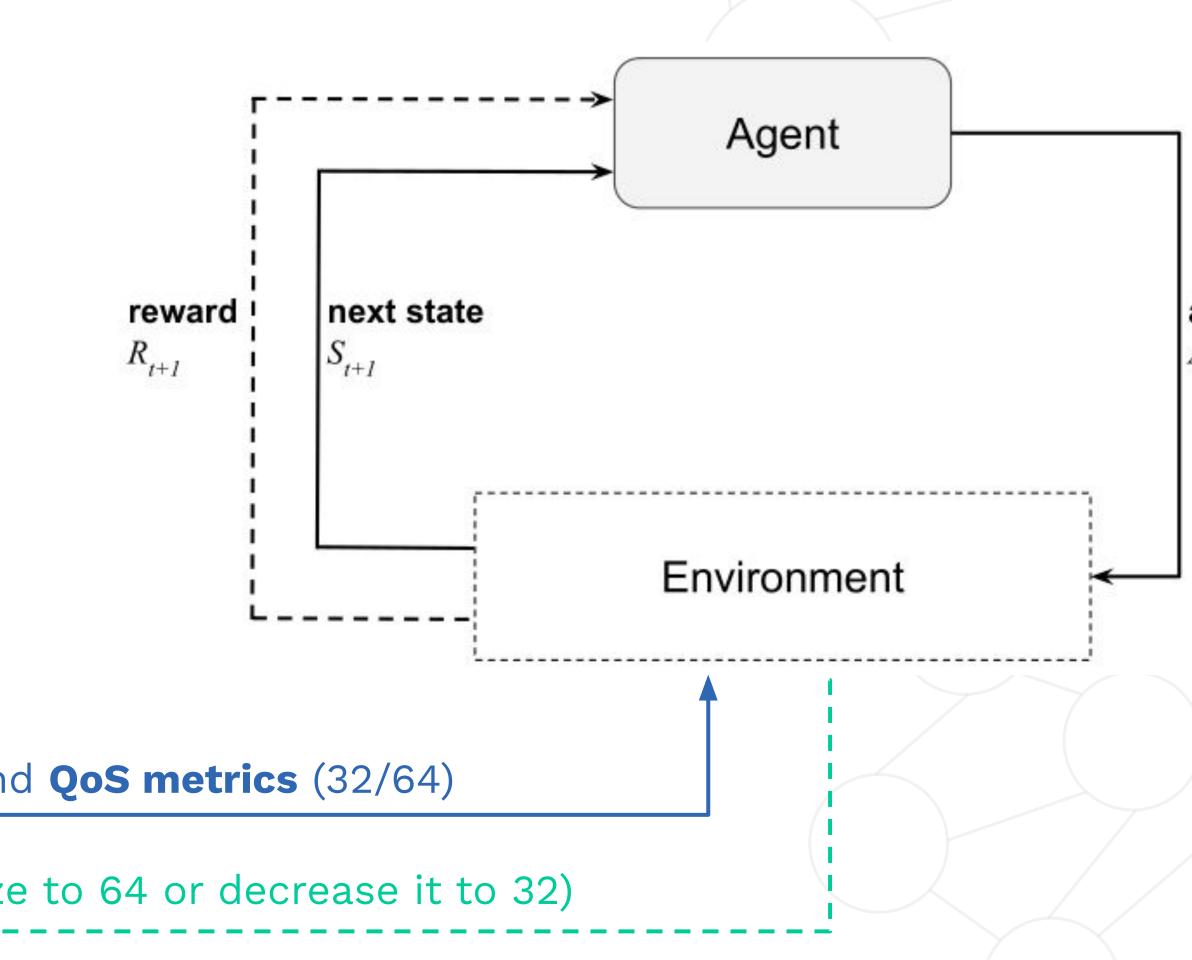
Synthetic data

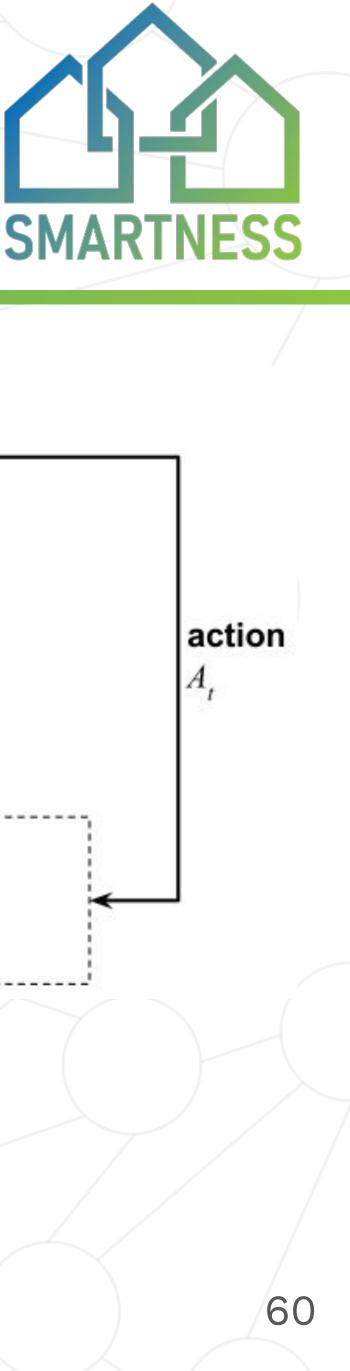
INT metadata (32/64) and QoS metrics (32/64)

action (increase the queue size to 64 or decrease it to 32)

receiveMetrics.py

3





Simulating the real network behavior

```
def readFile32():
   global sample32
    # Define the columns of interest
   cols = ['enq qdepth1', 'deq timedelta1', 'deq qdepth1',
            ' enq qdepth2', ' deq timedelta2', ' deq qdepth2',
            'enq qdepth3', 'deq timedelta3', 'deq qdepth3',
            'FPS', 'Buffer', 'CalcBitrate', 'ReportedBitrate']
    # Read the CSV file in chunks of 4 seconds
    for sample32 in pd.read_csv('synthetic_data/best_modelsum 32.csv', chunksize=4):
        # Select only the specified columns
        sample32 = sample32[cols]
        # Process the data using the 'jointoRL' function with TYPE 32
        jointoRL(sample32, TYPE 32)
 Start reading files in separate threads
thread64 = threading.Thread(target=readFile64)
thread64.start()
thread32 = threading.Thread(target=readFile32)
thread32.start()
```



Simulating the real network behavior

```
def jointoRL(sample, t):
   global sampleJoin
   # Store the data sample in the global dictionary using the specified type 't'
   sampleJoin[t] = sample
   # If two data samples have been collected (INT and DASH metrics related to the 32 and 64 queue sizes),
   # call the 'sendtoRl' function
   if len(sampleJoin) == 2:
       sendtoRl(sampleJoin)
```



Simulating the real network behavior

```
def sendtoRl(sample):
   global ddqn
   global env
   global experiment id
   # Verify whether the agent has already taken actions
   if len(env.actions history) == 0:
       # Determine the type of data sample and retrieve the DataFrame from the global dictionary accordingly
       if list(sample).index(0) == TYPE 32:
           df INT = sample[TYPE 32].iloc[:,:9]
           df dash = sample[TYPE 32].iloc[:,9:12]
       else:
           df INT = sample[TYPE 64].iloc[:,:9]
           df dash = sample[TYPE 64].iloc[:,9:12]
   # If the agent has already taken actions, verify which action was taken
   else:
       if env.actions history[-1] == 0:
           df INT = sample[TYPE 64].iloc[:,:9]
           df_dash = sample[TYPE_64].iloc[:,9:12]
       else:
```



Simulating the real network behavior

df_INT = sample[TYPE_32].iloc[:,:9]

df_dash = sample[TYPE_32].iloc[:,9:12]

Convert data to numpy arrays current state = df INT.to numpy() dash_state = df_dash.to_numpy()

Get state dimensionality state dim = df INT.shape[1]

Choose an action using epsilon-greedy policy action = ddqn.epsilon greedy policy(current state[FOURTH SECOND].reshape(-1, state dim))

Take the chosen action and observe the next state, reward, and done flag current_state, next_state, reward, done, _ = env.take_action(action, current_state, dash_state)

If next state is available, memorize the transition and perform experience replay if next state is not None:

print("next state received, memorizing transition")



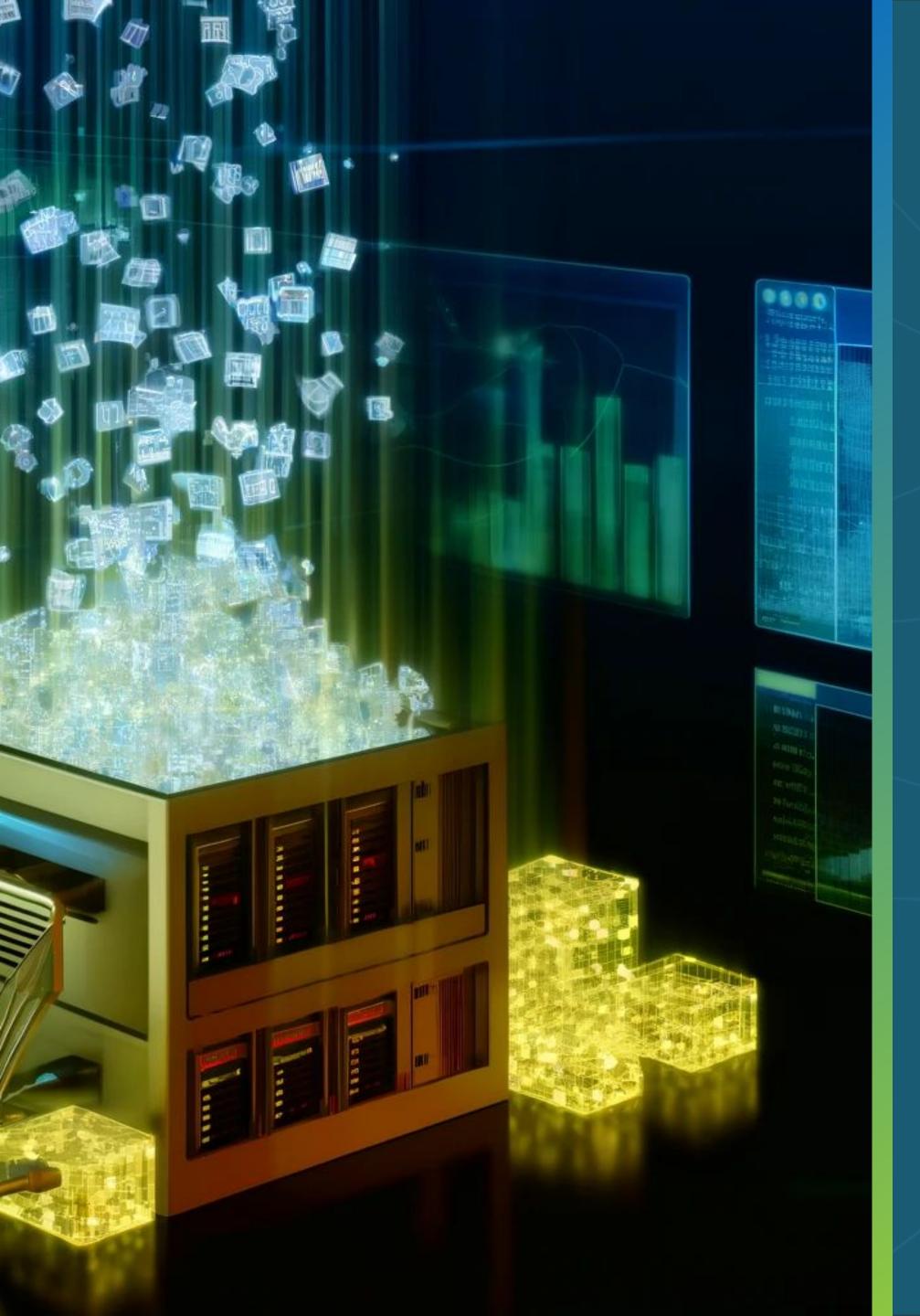


Simulating the real network behavior

```
ddqn.memorize transition(current state[FOURTH SECOND],
                     reward,
                     next_state[FOURTH_SECOND],
                     0.0 if done else 1.0)
   if ddqn.train:
      ddqn.experience replay()
print("last action: {0} | reward: {1} | fps: {2} | "
    "Buffer size: {3}".format(
     env.actions history[-1], env.reward history[-1],
     env.fps history[-1], env.buffer size[-1]))
```



env.actions_history[-2], # Action performed before reward calculation



Gen NetD Augm Proto

Jiang, Xi and Liu, Shinan and Gember-Jacobson, Aaron and Bhagoji, Arjun Nitin and Schmitt, Paul and Bronzino, Francesco and Feamster, Nick



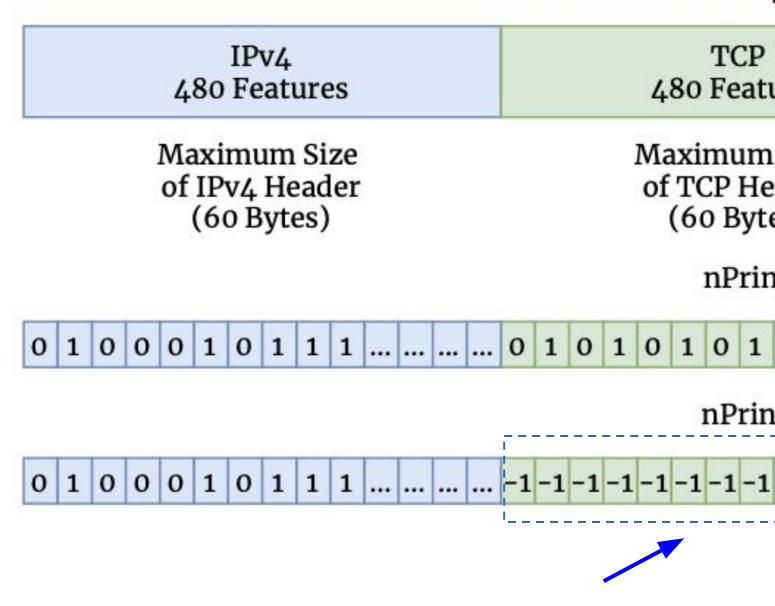
Generation of Synthetic Network Trace

NetDiffusion: Network Data Augmentation Through Protocol-Constrained Traffic Generation





nPrint Generation of Synthetic Network Trace



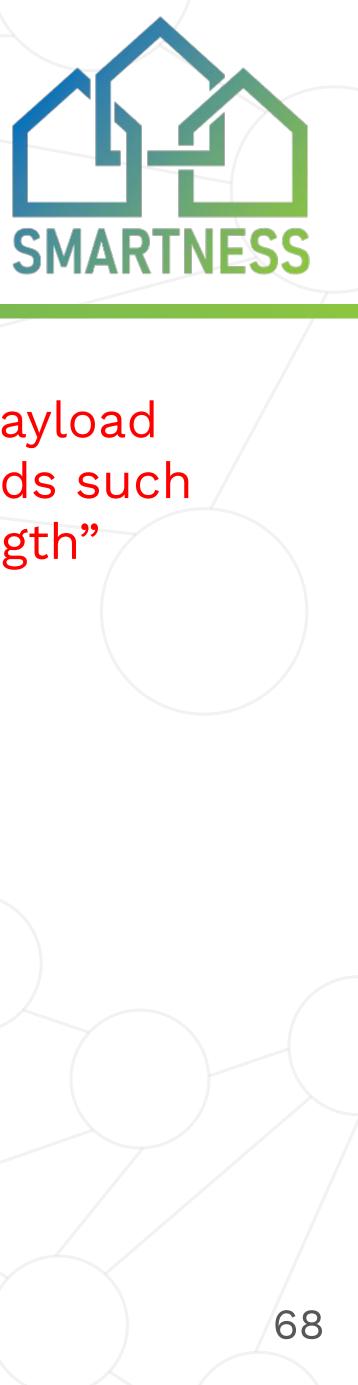


nPrint

e tures	UDP 64 Features	ICMP 64 Features	Payload n Features
n Size leader tes)	Size of UDP Header (8 Bytes)	Size of ICMP Header (8 Bytes)	User Defined Number of Bytes
int (TCP / IP) Pac	ket	<u> </u>	
l 0 1	-1 -1 -1 -1 -1 -1 -1	-1 -1 -1 -1 -1 -1 -1	1 1 0
nt (UDP / IP) Pac	ket		1
1 -1 -1 -1 -1 -1 -1	0 1 1	-1 -1 -1 -1 -1 -1 -1	1 1 0

nPrint Generation of Synthetic Network Trace

TCP 480 Featu				es		Pv4 Feat		4				
Maximum of TCP Hea (60 Byte		Maximum Size of IPv4 Header (60 Bytes)										
nPrint												
0 1 0 1 0 1 0 1	. 0 1 0	 		1	1	0 1	1	0	0	0	1	0
nPrint												
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -	1 -1 -	 		1	1	0 1	1	0	0	0	1	0

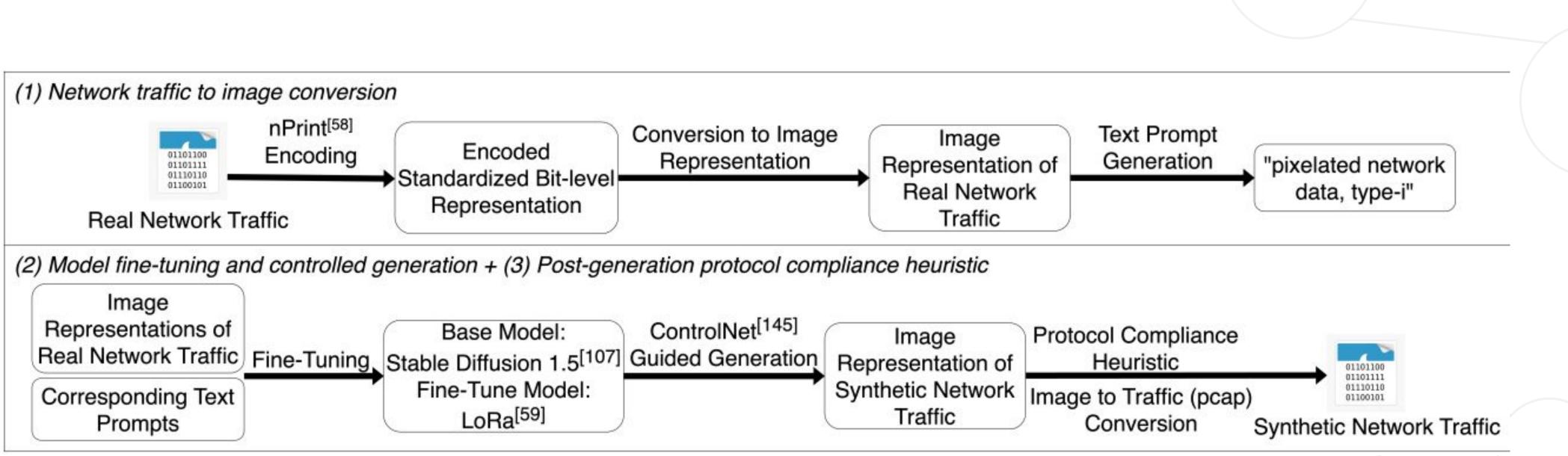


We can be infer the payload length from header fields such as the IP "Total Length"

nPrint Payload UDP ICMP **64 Features 64** Features n Features ures Size Size of Size of User Defined ICMP Header **UDP Header** eader Number of Bytes es) (8 Bytes) (8 Bytes) nt (TCP / IP) Packet nt (UDP / IP) Packet -1 -1 -1 -1 -1 -1 0 1 1 -1 -1 -1 -1 -1 -1 -1 1 1 0

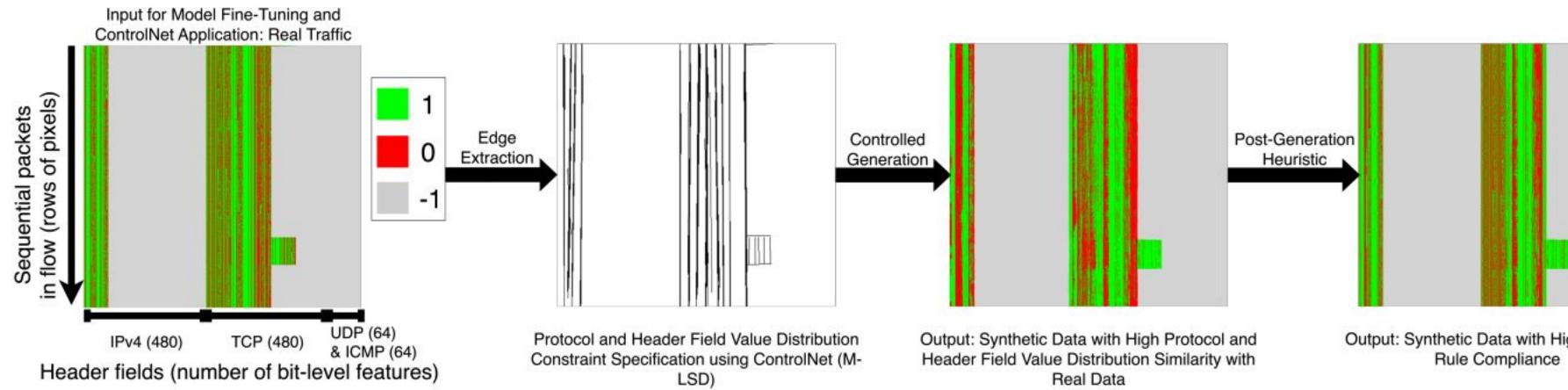
NetDiffusion: Workflow

Generation of Synthetic Network Trace





NetDiffusion: Workflow (Example) Generation of Synthetic Network Trace



Synthetic Amazon network traffic outputs: (1) Using ControlNet, it detect regions present in the original traffic and ensure protocol and header field value distribution conformance by generating within specified regions. (2) Applying a post-generation heuristic to refine field details for protocol conformance.

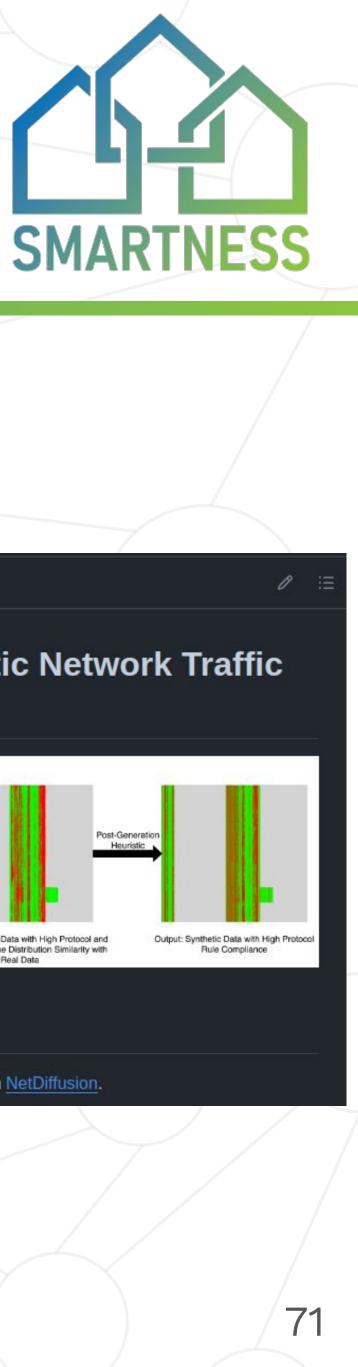


Output: Synthetic Data with High Protocol

Hands on

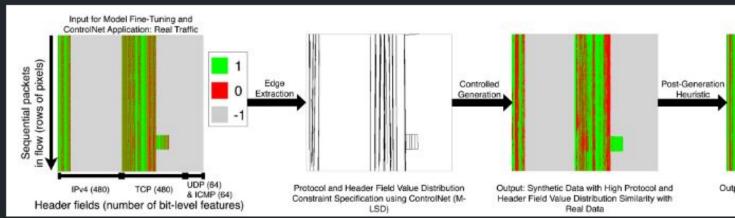
Generation of Synthetic Network Trace

GitHub <u>https://github.com/arielgoes/NetDiffu</u>



C README A License

NetDiffusion: High-Fidelity Synthetic Network Traffic Generation



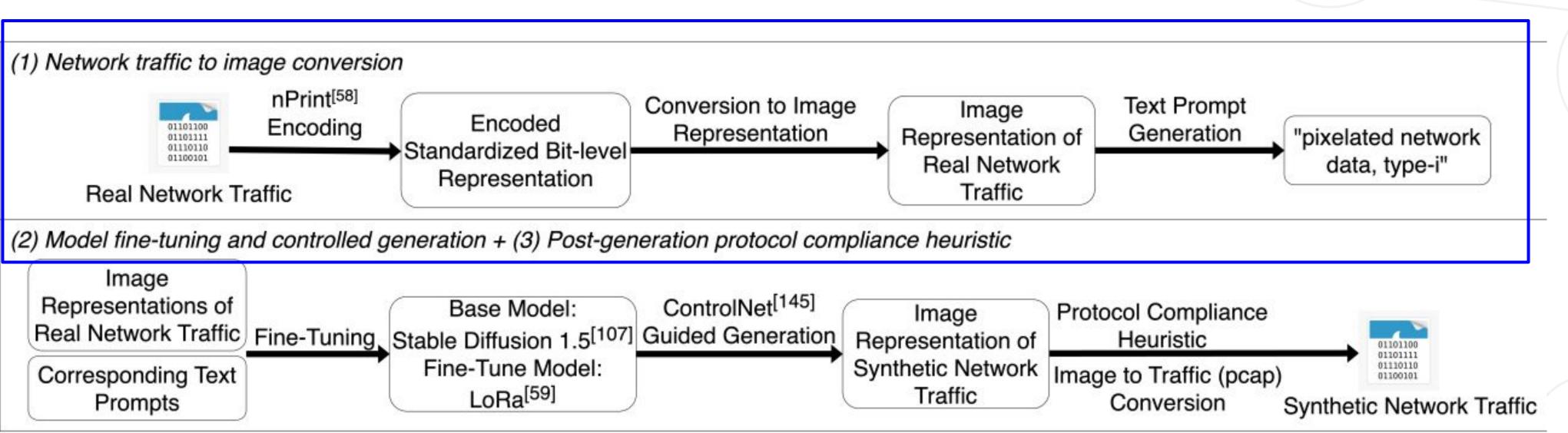
NetDiffusion tutorial

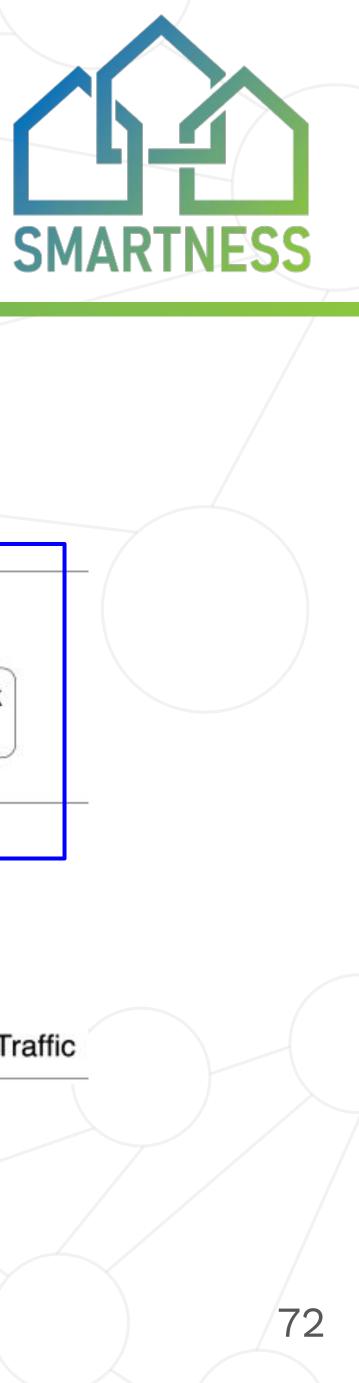
This tutorial is based on the original guide steps for generating PCAPs with NetDiffusion.



NetDiffusion: Workflow

Generation of Synthetic Network Trace

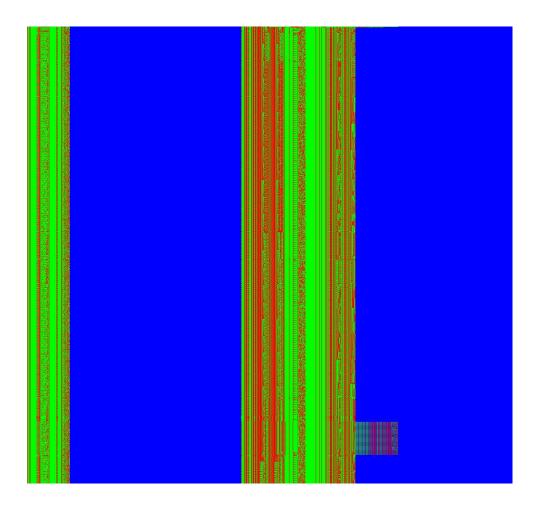




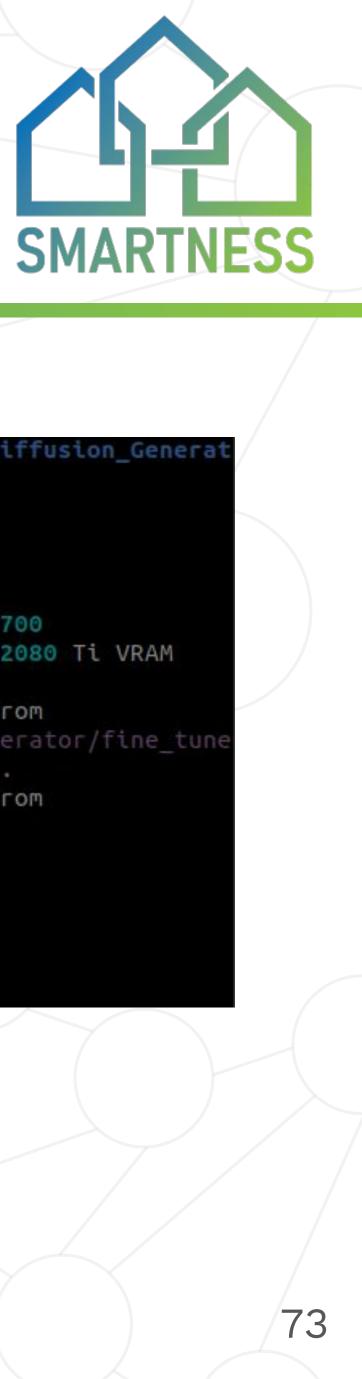
Steps 1-4

NetDiffusion: Main Steps Generation of Synthetic Network Trace

1. Input converted nPrint to image (1088x1024):



blue = -1 green = 1 red = 0 **caption:** *"pixelated* network data, type-0"

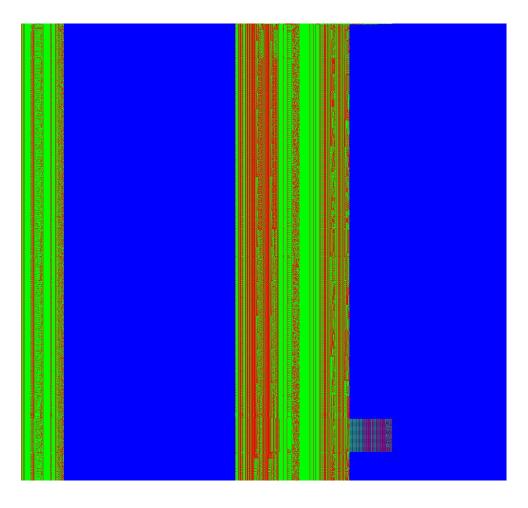


2. Accessing the GUI

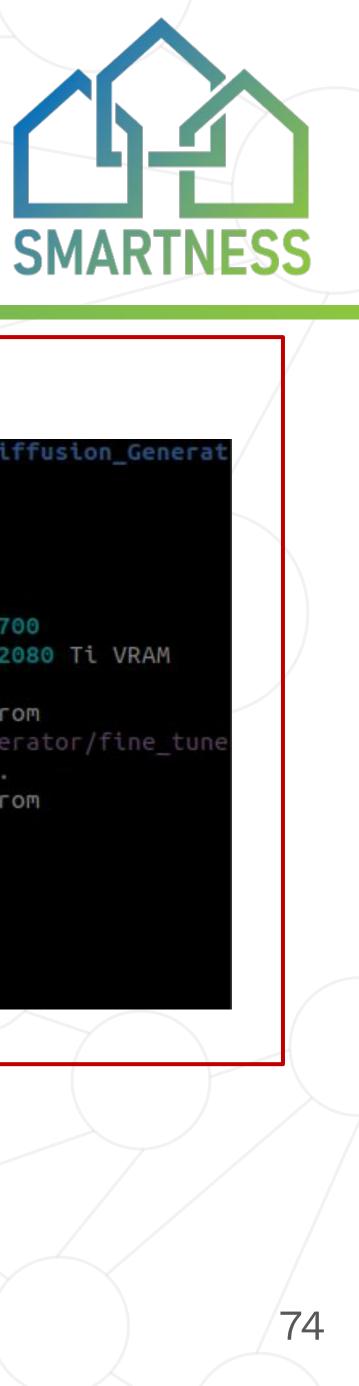
venv) (base) thiago@ifsuldeminas-Z390-M-GAMING:~/git_ariel/NetDiffusion_Generat or/fine_tune/kohya_ss_fork\$./gui.sh :24:52-270585 INFO Version: v22.6.2 nVidia toolkit detected :24:52-288388 INFO Torch 2.0.1+cu118 :24:53-107035 INFO Torch backend: nVidia CUDA 11.8 cuDNN 8700 :24:53-118370 INFO Torch detected GPU: NVIDIA GeForce RTX 2080 Ti VRAM :24 53-128552 INFO 11009 Arch (7, 5) Cores 68 2:24:53-129438 INFO Verifying modules installation status from /home/thiago/git_ariel/NetDiffusion_Generator/fine_tune /kohya_ss_fork/requirements_linux.txt... :24:53-131247 INFO Verifying modules installation status from requirements.txt... headless: False 2:24:55-141001 INFO 2:24:55-143411 INFO Load CSS... Running on local URL: http://127.0.0.1:7860 To create a public link, set `share=True` in `launch()`.

Generation of Synthetic Network Trace

1. Input converted nprint to image (1088x1024):



blue = -1 green = 1 red = 0**caption:** *"pixelated* network data, type-0"

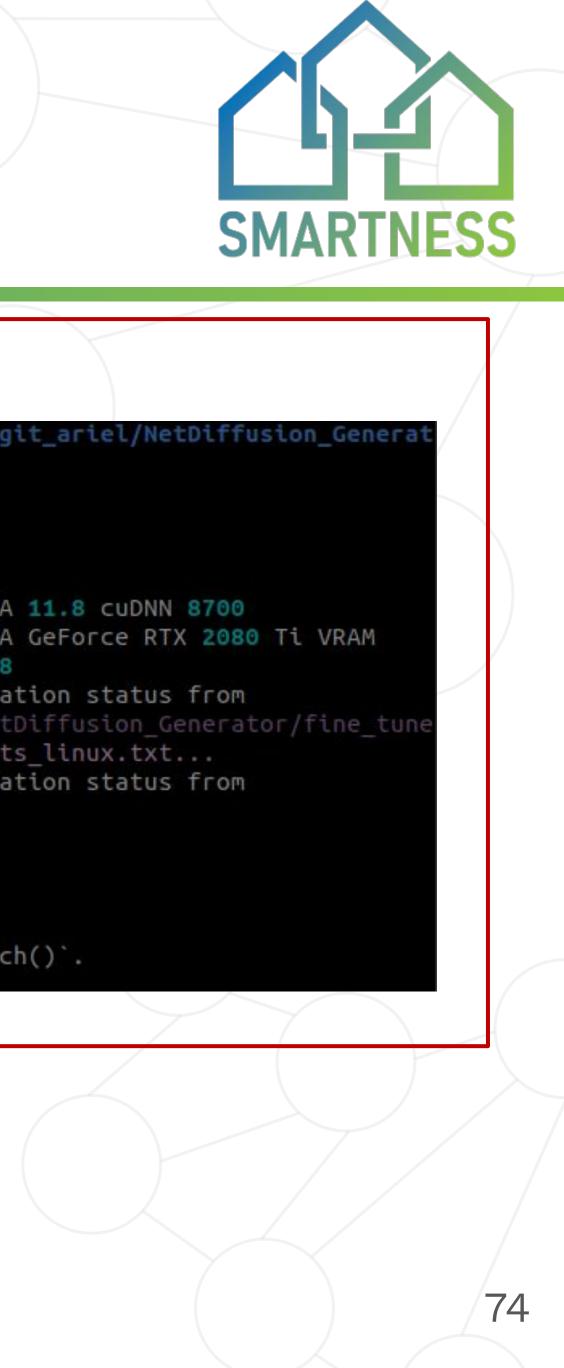


2. Accessing the GUI

(base) thiago@ifsuldeminas-Z390-M-GAMING:~/git_ariel/NetDiffusion_General r/fine_tune/kohya_ss_fork\$./gui.sh Version: v22.6.2 4:52-270585 INFO

12:24:52-288388	INFO	nVidia toolkit detected
12:24:53-107035	INFO	Torch 2.0.1+cu118
12:24:53-118370	INFO	Torch backend: nVidia CUDA 11.8 cuDNN 8700
12:24:53-128552	INFO	Torch detected GPU: NVIDIA GeForce RTX 2080 Ti VRAM
		11009 Arch (7, 5) Cores 68
12:24:53-129438	INFO	Verifying modules installation status from /home/thiago/git_ariel/NetDiffusion_Generator/fine_tur /kohya_ss_fork/requirements_linux.txt
12:24:53-131247	INFO	Verifying modules installation status from requirements.txt
12:24:55-141001	INFO	headless: False
12.24:55-143411	INFO	Load CSS
Running on local	l URL:	http://127.0.0.1:7860

create a public link, set `share=True` in `launch()`.



Generation of Synthetic Network Trace

3.	Fine-tuning	image/model/lo	g paths
----	-------------	----------------	---------

Dreambooth LoRA Textual Inversion Finetuning Utilities About	
Training Tools Guides	
Train a custom model using kohya train network LoRA python code	
Configuration file	•
Source model Folders Parameters Dataset Preparation	
Image folder /home/thiago/git_ariel/NetDiffusion_Generator/fine_tune/kohya_ss_fork/model_training/test_task/ima ge	Regularisation folder (Optional) Folder where where the regularization folders containing the images are located
Output folder /home/thiago/git_ariel/NetDiffusion_Generator/fine_tune/kohya_ss_fork/model_training/test_task/mo del	Logging folder /home/thiago/git_ariel/NetDiffusion_Generator/fine_tune/kohya_ss_fork/model_training/test_task/log
Model output name	Training comment
last	(Optional) Add training comment to be included in metadata
Start training	Stop training
Print trainin	ng command
Start tensorboard	Stop tensorboard
Textbox	





Train a custom model using kohya train netwo	ork LoRA python code			
Configuration file				
Source model Folders Paramete	Dataset Preparation			
Presets				
none	•			
Basic Advanced Samples				
LoRA type				
Standard				
Train batch size	Epoch	Max train epoch		
	1	(Optional) Enforce number of		
Mixed precision	Save precision	Number of CPU 2 threads per core		
fp16 +	fp16 •			
LR Scheduler				
cosine		:		
Max grad norm	1	LR scheduler extra arguments		
	C	(Optional) eg: "milestones=[1,10,30,5		
Learning rate				
			2	
LR number of cycles				
Max resolution				
816,768				



SMAR

Generation of Synthetic Network Trace

3. Fine-tuning image/model/log paths

Dreambooth LoRA Textual Inversion Finetuning Utilities About	
Training Tools Guides	
Train a custom model using kohya train network LoRA python code	
Configuration file	•
Source model Folders Parameters Dataset Preparation	
Image folder /home/thiago/git_ariel/NetDiffusion_Generator/fine_tune/kohya_ss_fork/model_training/test_task/ima ge	Regularisation folder (Optional) Folder where where the regularization folders containing the images are located
Output folder /home/thiago/git_ariel/NetDiffusion_Generator/fine_tune/kohya_ss_fork/model_training/test_task/mo del //	Logging folder /home/thiago/git_ariel/NetDiffusion_Generator/fine_tune/kohya_ss_fork/model_training/test_task/log
Model output name	Training comment
last	(Optional) Add training comment to be included in metadata
Start training	Stop training
Print traini	ng command
Start tensorboard	Stop tensorboard
Textbox	





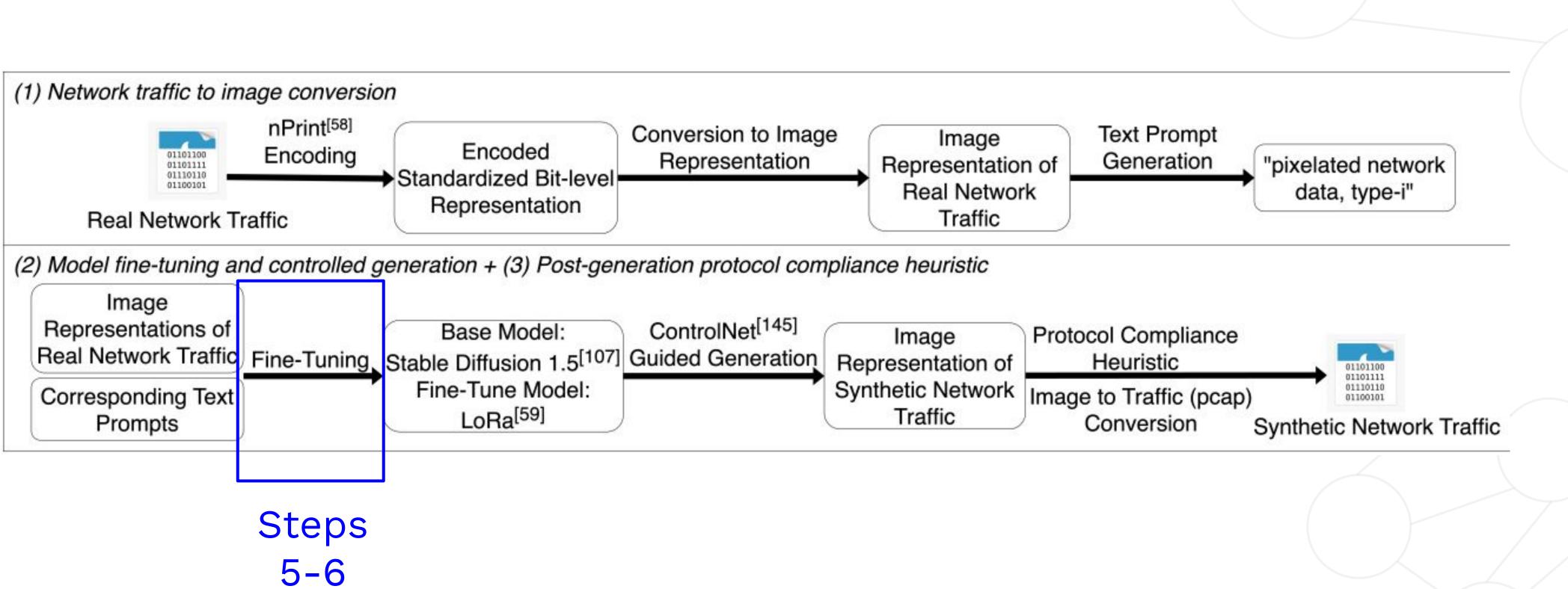
Tain a cotom model using holys train network LoRA python code Configuration file Source model fieldes Parameters Extra del Paraportion Pessis none CodeA type: Savedard Fains batch hize	Dreambooth LoRA Textual Inversion Finetuning Utilities About Training Tools Guides
Source model Folders Presets none Basic Marcod Songles LoPA tope Standard Initial basic basic I Epsch Marcod Songles LoPA tope Standard Initial basic basic I Ippedia Marcod Initial basic basic I Image precision Image preci	Train a custom model using kohya train network LoRA python code
Pressis none Basic Advanced Samples Look type Sandard Finin batch size 1 Cyclonal) Enforces number of Mixed precision Mixed precision Fig16 ER Scheduler cosine R scheduler Cyclonal) egf-nonests 1 Defining cole noosil It Scheduler Cyclonal) egf-nonests 1 Defining cole noosil Interder of cycles Cyclonal) Efficient with restart and polynomial city Nar resolution	Configuration file
mm Basic Advanced Samples LeRA type Sandard Tain batch size In batch size <	Source model Folders Dataset Preparation
LsBà type Sandard Train batch size 1 Epoch Mas train epoch 1 Opponall Enforce number of Mixed prechion Save preciaion Number of CFU 2 Mixed prechion Save preciaion Number of CFU 2 (p15 (p16 (p16 (p16))) LR Scheduler extra arguments (Opponall eg' 'milestones:1,1,0,1c) Learning rate 0,0001 Learning rate 0,0001 Learning rate (Opponall For Cosine with restart and polynomial enty Max resolution	
Standard Train batch size I Epoch Mixed precision Number of CPU Ip16 Ip16 In Coptionally Extreme It Scheduler It Scheduler <td>Basic Advanced Samples</td>	Basic Advanced Samples
Train batch size 1 Epoch Max train epoch (Optional) Enforce number of Mixed precision Save precision Number of CPU (2) #p16 (2) #p16 (2) #p16 (2) #p16 (2) (2) (2) (2) (2) (2) (2) (2) (2) (2) (2) (2) (2) (2) (2) (3) (2) (2) (3) (2) (4) (2) (2) (2) (2) (3) (3) (4	LoRA type
I (Poptional) Enforce number of cPU Mixed precision Save precision Typ16 ************************************	Standard
1 (ppional) Enforce number of procession Mixed precision Save precision Number of CPU 2 fp16 1	
Mixed precision Save precision Number of CPU 2 fp16 • fp16 • fp16 • Cosine	
Image: type of type	
fp16 fp16 LR Scheduler cosine Max grad norm 1 LR scheduler extra arguments (Optional) eg: "milestones=[1,10,30] Learning rate 0.0001 LR number of cycles (Optional) For Cosine with restart and polynomial only	
cosine Max grad norm 1 Max grad norm 1 Coptional) eg: "milestones=[1,10,30] Learning rate 0.0001 LR number of cycles (Optional) For Cosine with restart and polynomial only Max resolution	
Max grad norm I LR scheduler extra arguments (Optional) eg: "milestones=[1,10,30.5 Learning rate 0.0001 LR number of cycles (Optional) For Cosine with restart and polynomial only Max resolution	LR Scheduler
(Optional) eg: "milestones=[1,10,30,5 Learning rate 0.0001 LR number of cycles (Optional) For Cosine with restart and polynomial only Max resolution	cosine
Learning rate 0.0001 LR number of cycles (Optional) For Cosine with restart and polynomial only Max resolution	Max grad norm 1 LR scheduler extra arguments
0.0001 LR number of cycles (Optional) For Cosine with restart and polynomial only Max resolution	(Optional) eg: "milestones=[1,10,30,5
LR number of cycles (Optional) For Cosine with restart and polynomial only Max resolution	Learning rate
(Optional) For Cosine with restart and polynomial only Max resolution	0.0001
Max resolution	LR number of cycles
	Max resolution
816, (65	816,768



SMAR

NetDiffusion: Workflow

Generation of Synthetic Network Trace



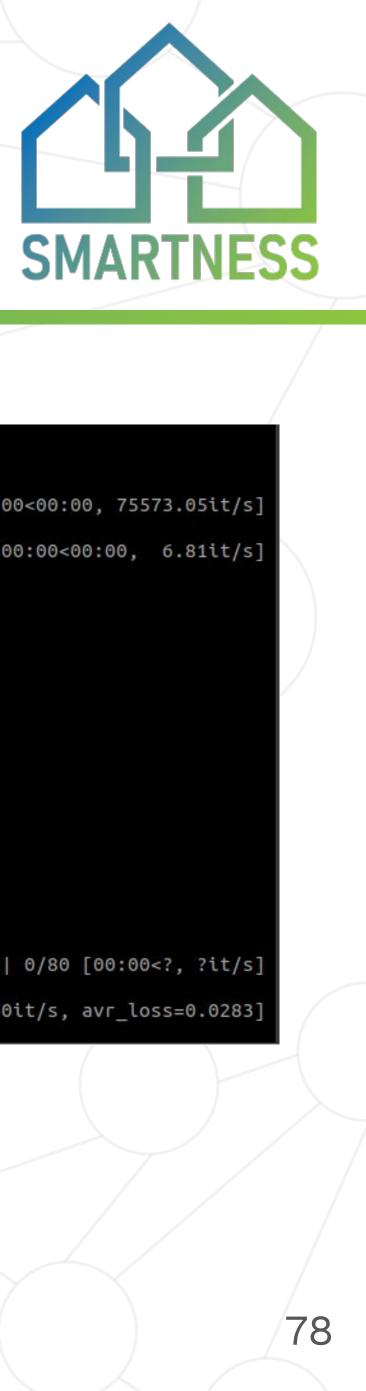


Generation of Synthetic Network Trace

5. Training the fine-tuned image

[Dataset 0] caching latents. checking cache validity... 00% aching latents... 00% reate LoRA network. base dim (rank): 8, alpha: 1.0 neuron dropout: p=None, rank dropout: p=None, module dropout: p=None reate LoRA for Text Encoder: reate LoRA for Text Encoder: 72 modules. create LoRA for U-Net: 192 modules. enable LoRA for text encoder enable LoRA for U-Net prepare optimizer, data loader etc. use 8-bit AdamW optimizer | {} running training / 学習開始 num train images * repeats / 学習画像の数×繰り返し回数: 80 num reg images / 正則化画像の数: 0 num batches per epoch / 1epochのバッチ数: 80 num epochs / epoch数: 1 batch size per device / バッチサイズ: 1 gradient accumulation steps / 勾配を合計するステップ数 = 1 total optimization steps / 学習ステップ数: 80 steps: 0% epoch 1/1 steps: 15%





4/4 [00:00<00:00, 75573.05it/s]

| 4/4 [00:00<00:00, 6.81it/s]

| 12/80 [00:04<00:27, 2.50it/s, avr loss=0.0283]



Generation of Synthetic Network Trace

6. Showing the image caption

(base) thiago@ifsuldeminas-Z390-M-GAMING:~/git_ariel/NetDiffusion_Generator/fine_tune/kohya_ss_fork/model_training/test_task/image/20_network\$ ls discord_01.png discord_01.txt netflix_01_1.png netflix_01_1.txt netflix_01_2.png netflix_01_2.txt netflix_01.png netflix_01.txt (venv) (base) thiago@ifsuldeminas-Z390-M-GAMING:~/git_ariel/NetDiffusion_Generator/fine_tune/kohya_ss_fork/model_training/test_task/image/20_network\$ cat netflix 01.txt pixelated network data, type-0 (venv) (base) thiago@ifsuldeminas-Z390-M-GAMING:~/git_ariel/NetDiffusion_Generator/fine_tune/kohya_ss_fork/model_training/test_task/image/20_network\$

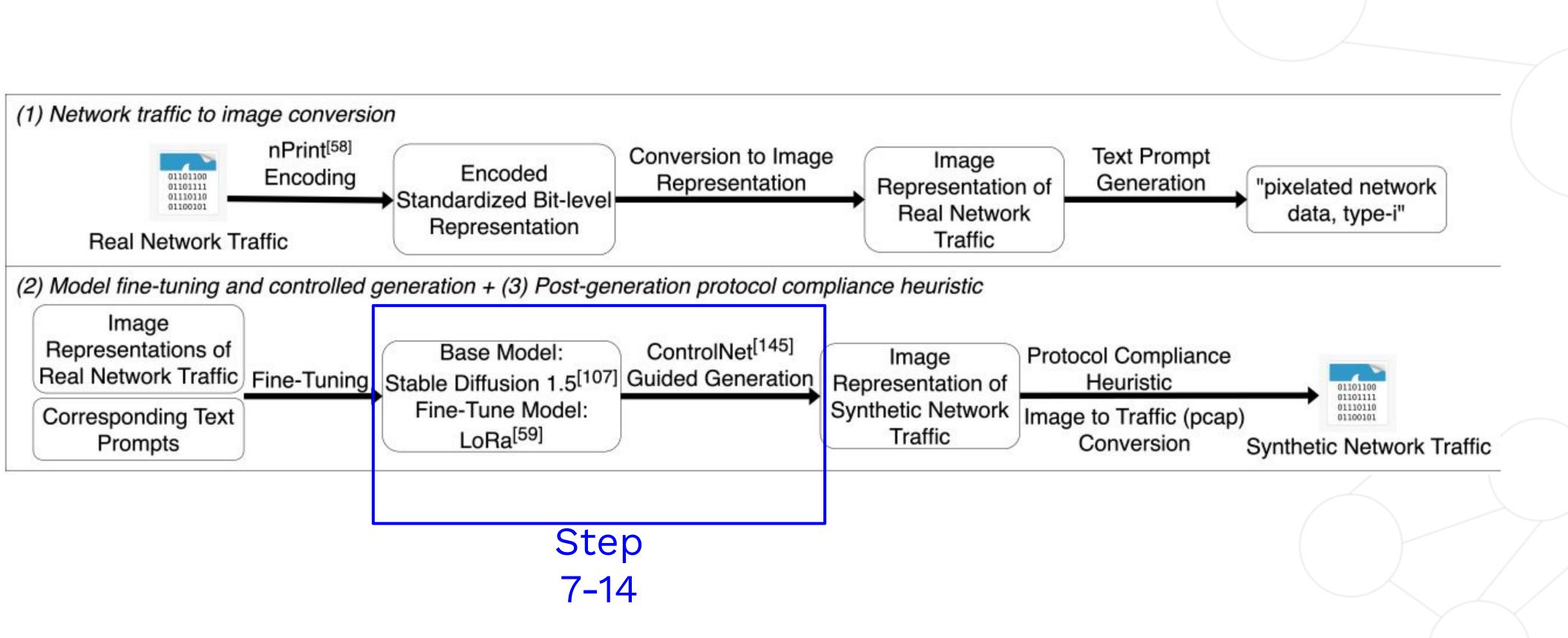






NetDiffusion: Workflow

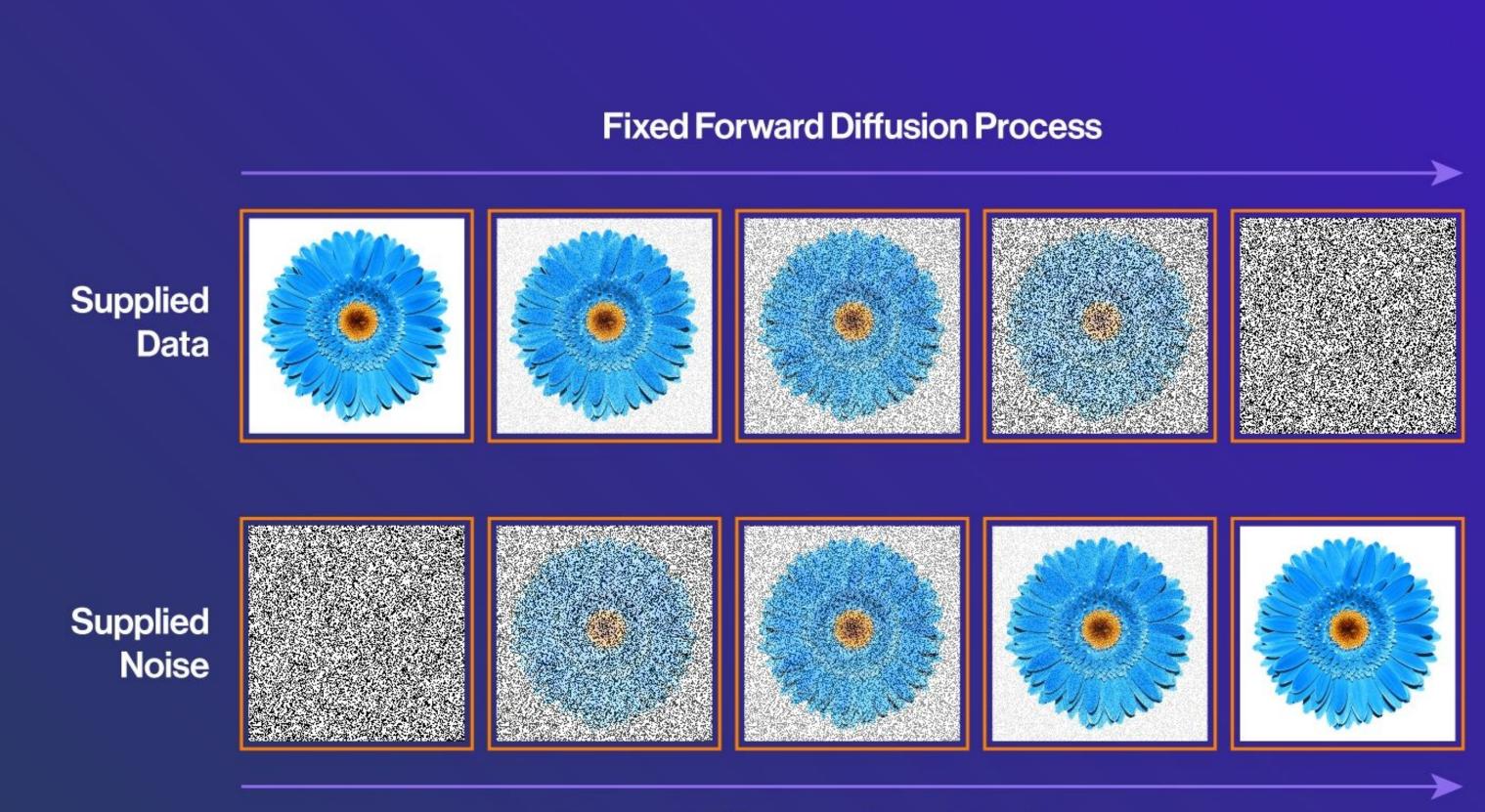
Generation of Synthetic Network Trace





What is a diffusion model (in Generative Als)?

Generation of Synthetic Network Trace



Generative Reverse Denoising Process



Generated Data

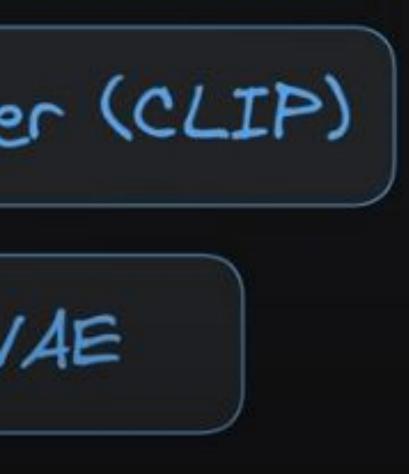


What is Stable Diffusion? Generation of Synthetic Network Trace

Main components:

U-Net	Text Encode
Noise Se	cheduler



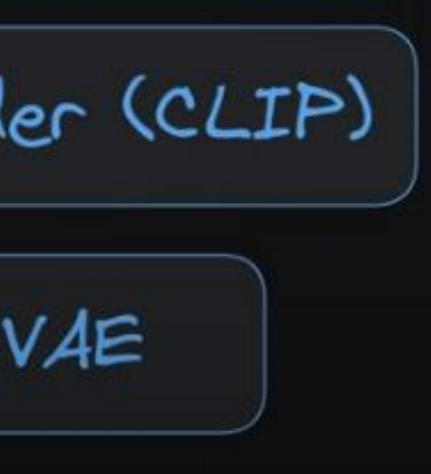


What is Stable Diffusion? Generation of Synthetic Network Trace

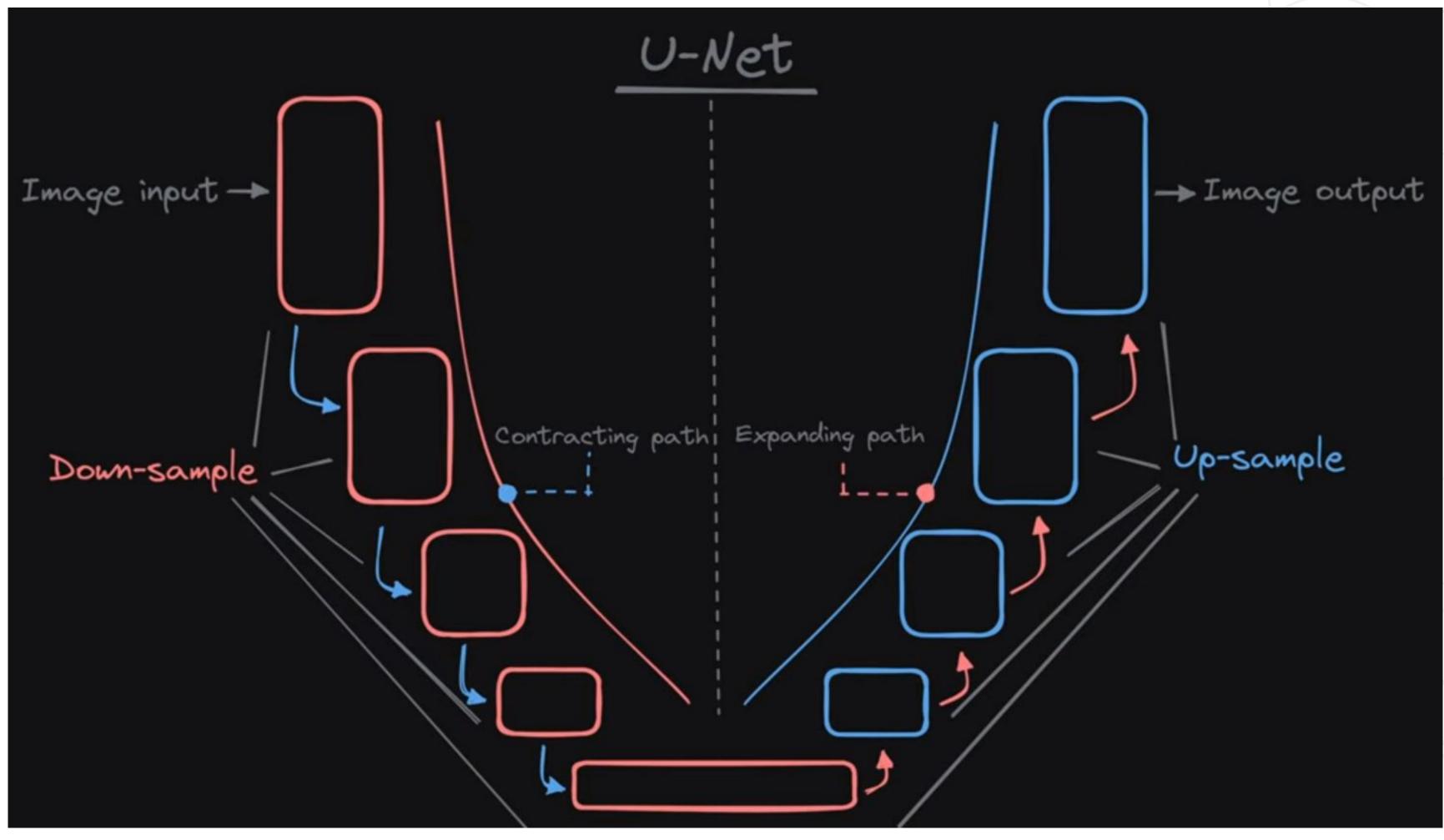
Main components:

Text Encoder (CLIP) U-Net Noise Scheduler





Stable Diffusion: U-Net Generation of Synthetic Network Trace

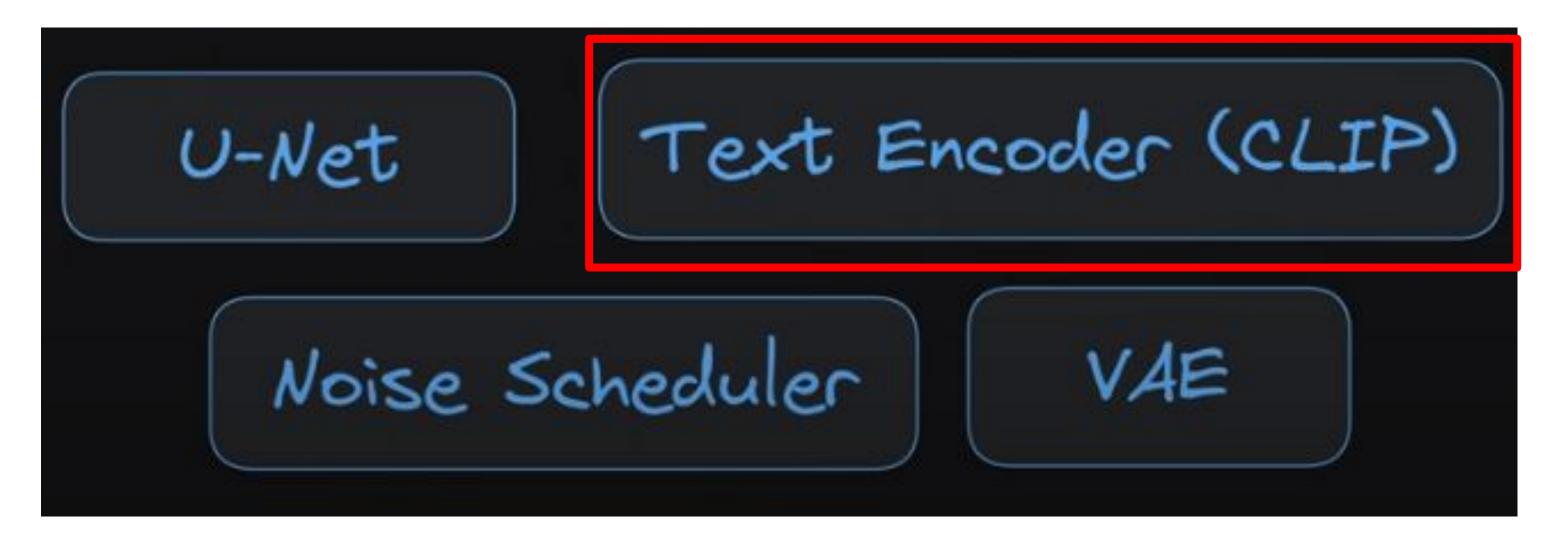




Stable Diffusion: Text Encoder (CLIP)

Generation of Synthetic Network Trace

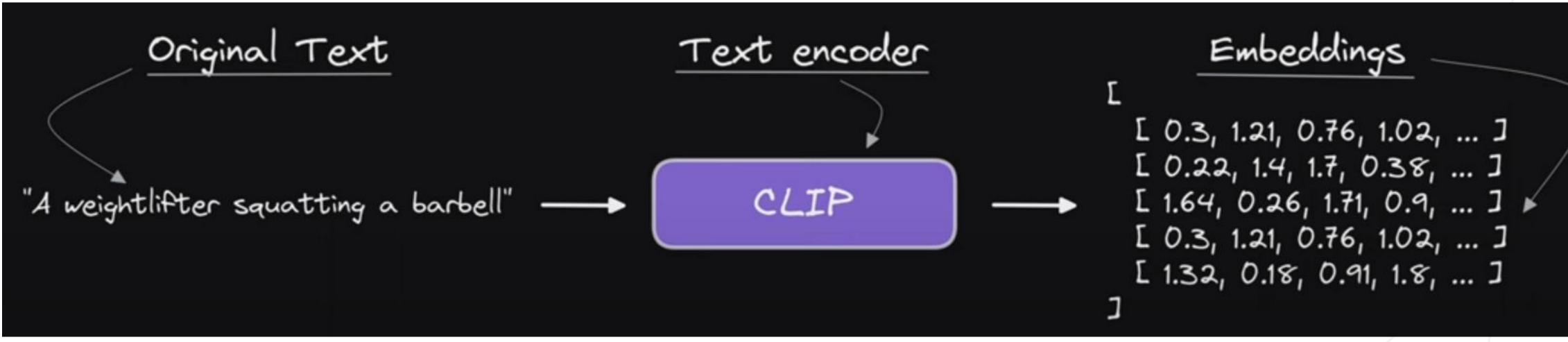
Main components:





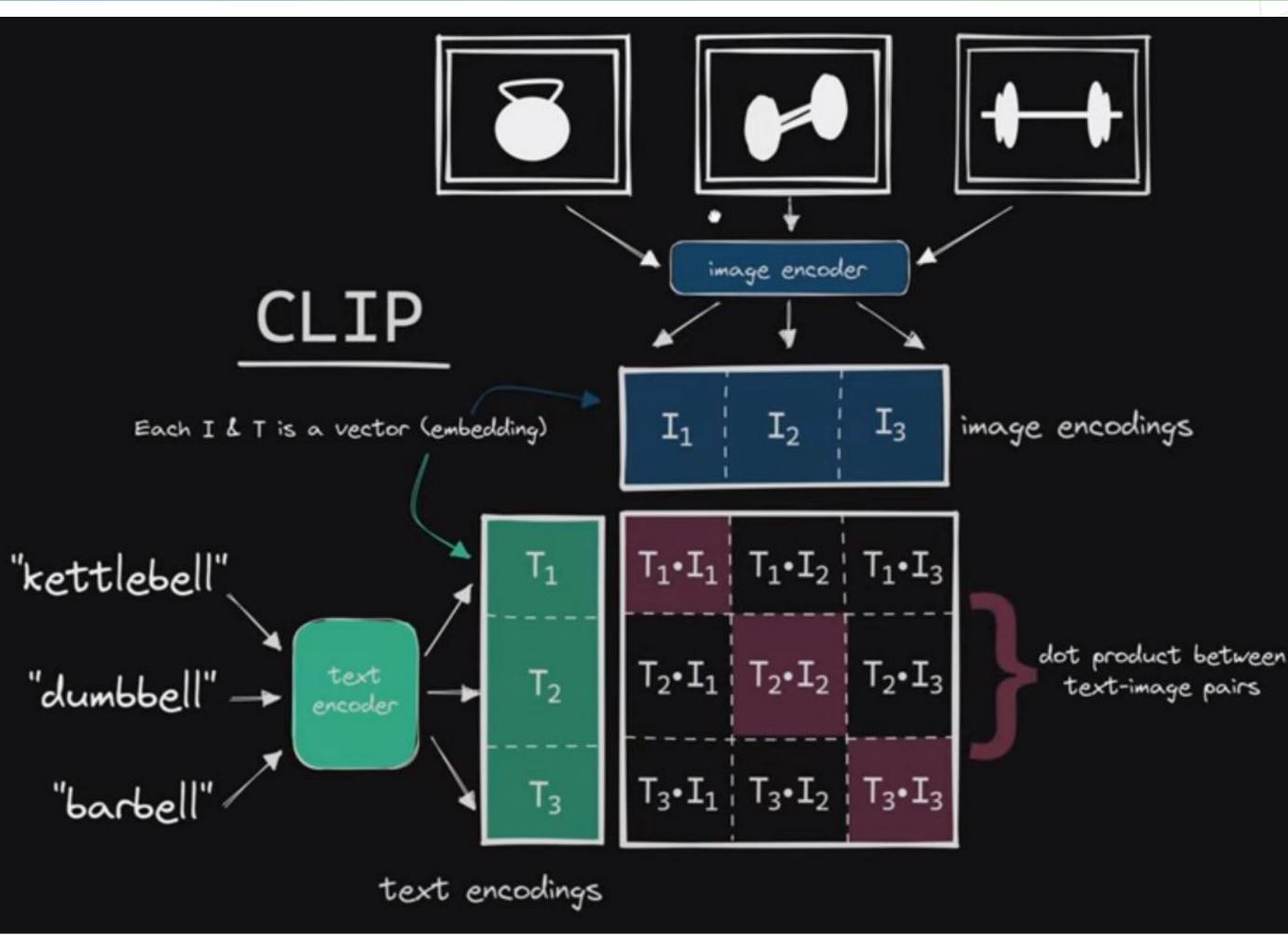
Stable Diffusion: Text Encoder (CLIP)

Generation of Synthetic Network Trace





Stable Diffusion: Text Encoder (CLIP) Generation of Synthetic Network Trace





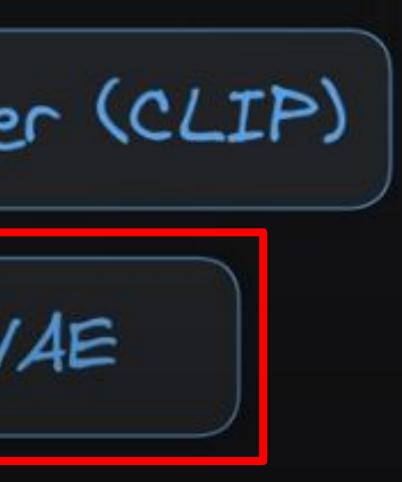
Stable Diffusion: Variational Autoencoder (VAE)

Generation of Synthetic Network Trace

Main components:

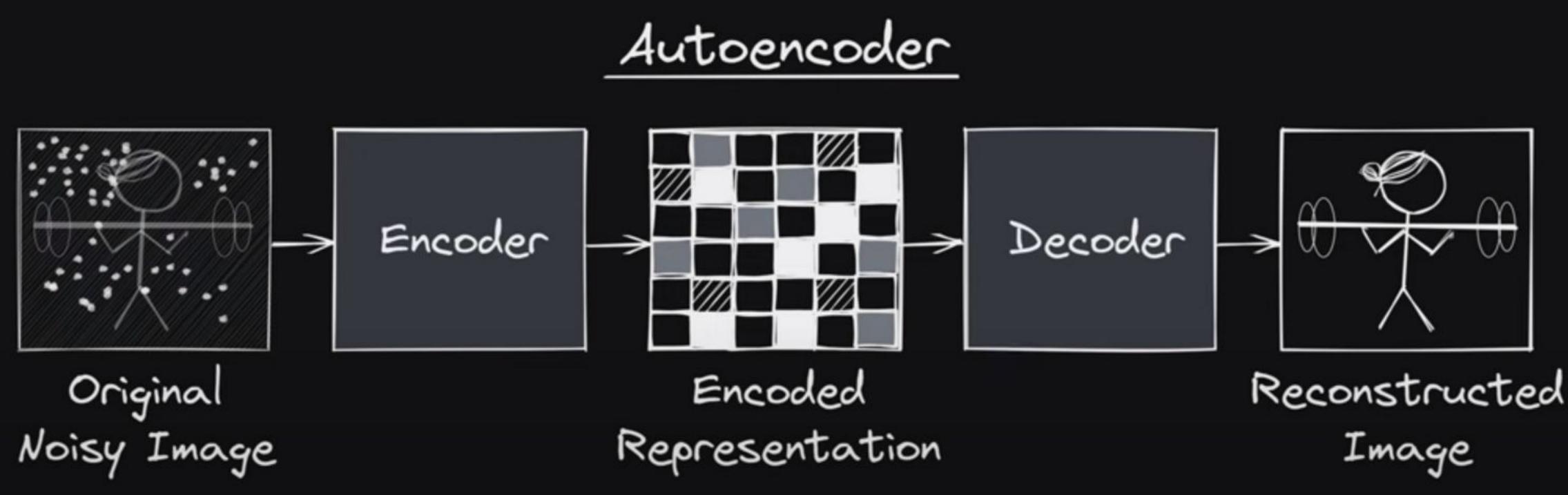
U-Net	Text Encode
Noise S	cheduler



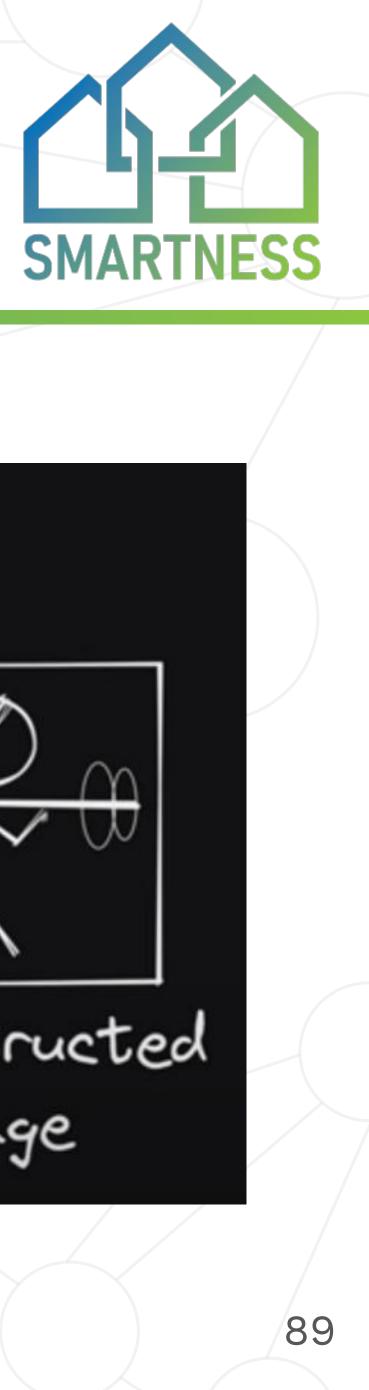


Stable Diffusion: Variational Autoencoder (VAE)

Generation of Synthetic Network Trace







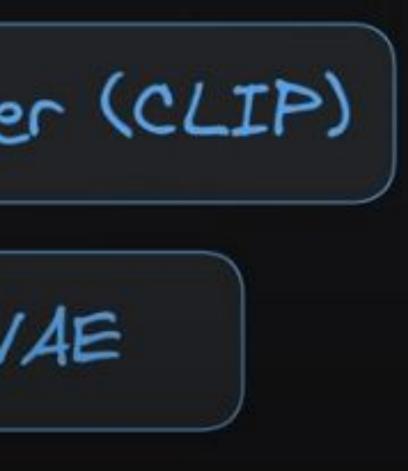
Stable Diffusion: Noise Scheduler

Generation of Synthetic Network Trace

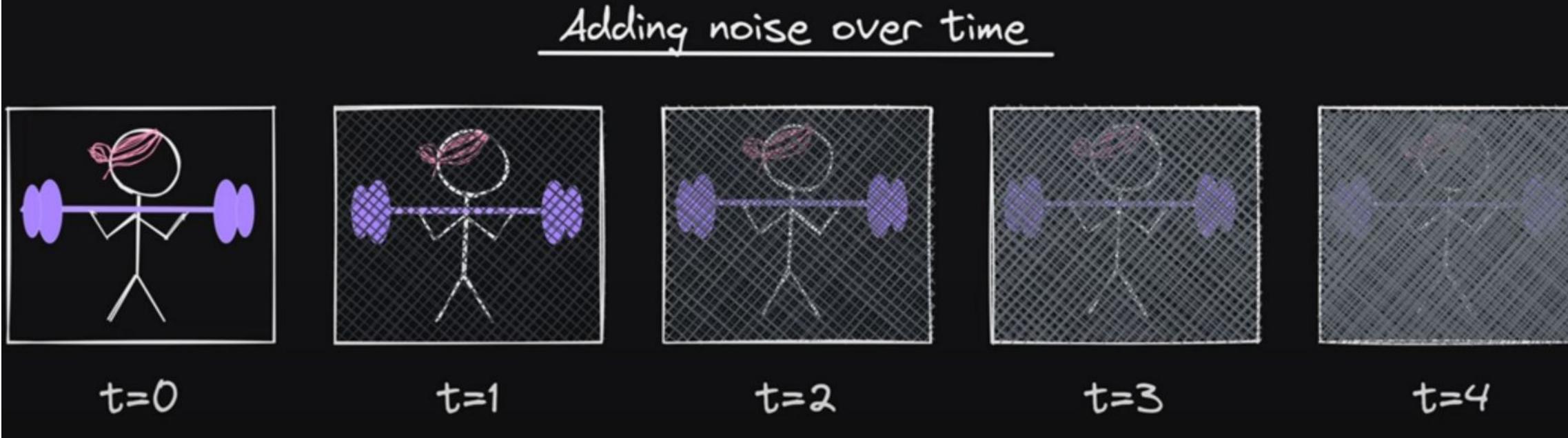
Main components:

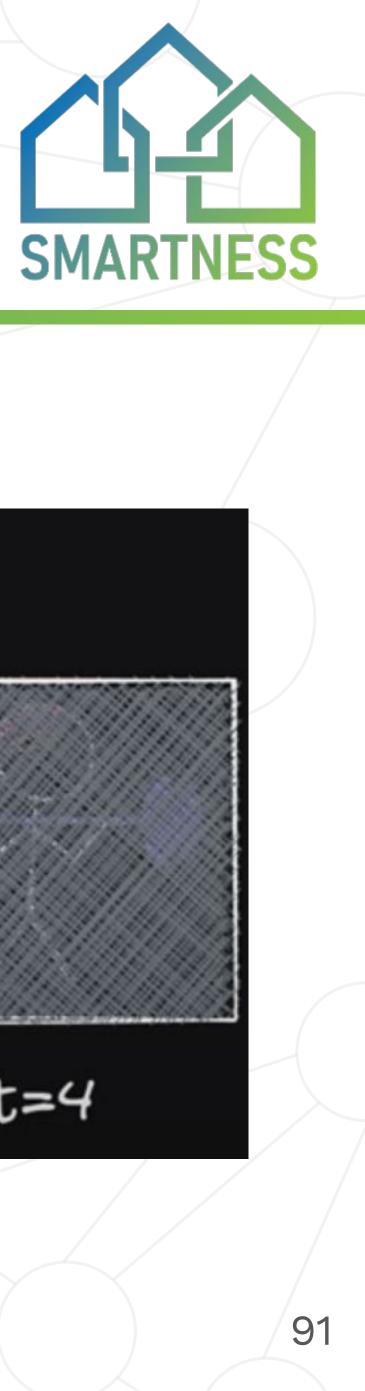
U-Net Text End	code
Noise Scheduler	



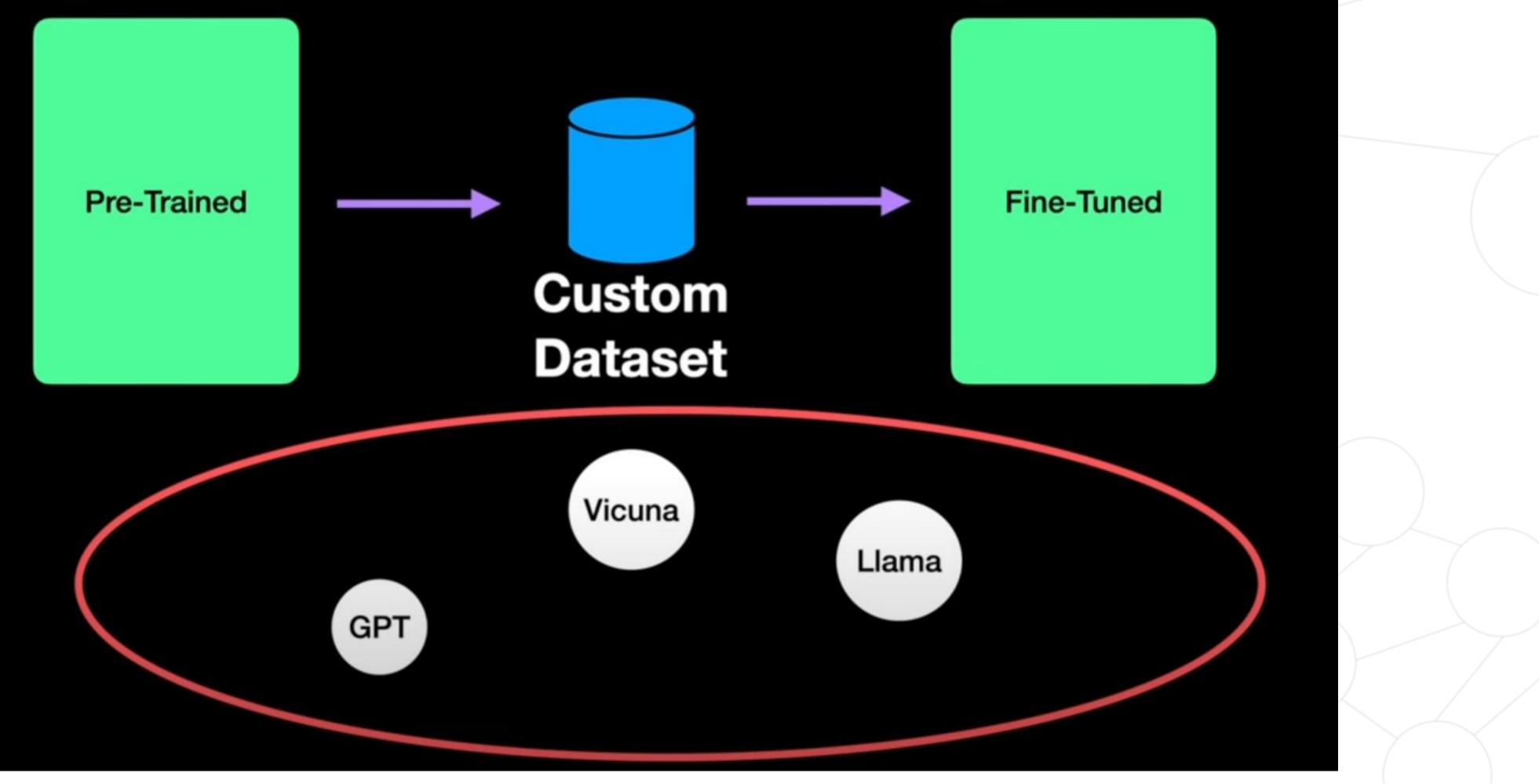


Stable Diffusion: Noise Scheduler Generation of Synthetic Network Trace





What is Low-Rank Adaptation (LoRa)? Generation of Synthetic Network Trace



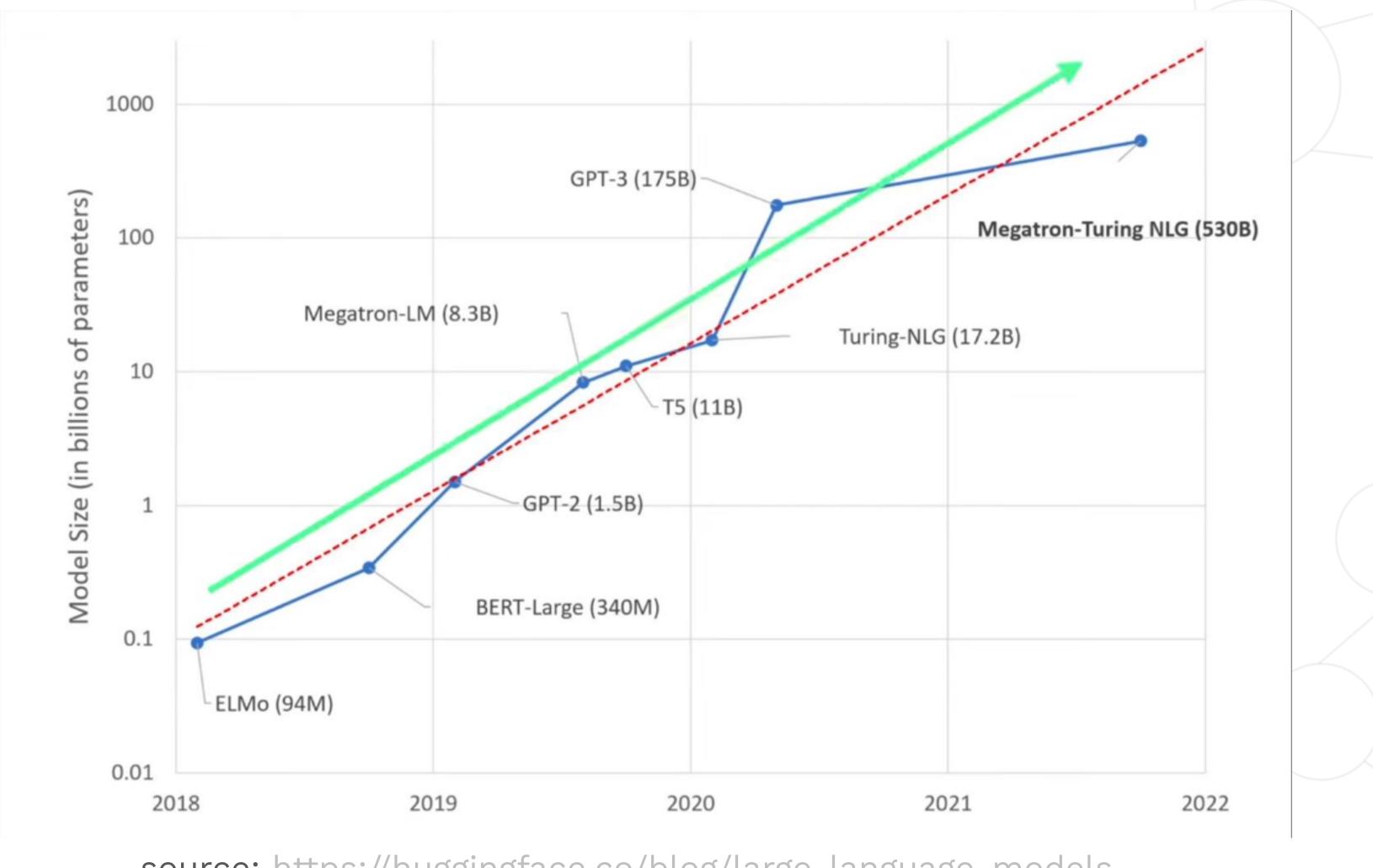
source: <u>https://www.youtube.com/watch?v=X4VvO3G6_vw</u>





What is Low-Rank Adaptation (LoRa)?

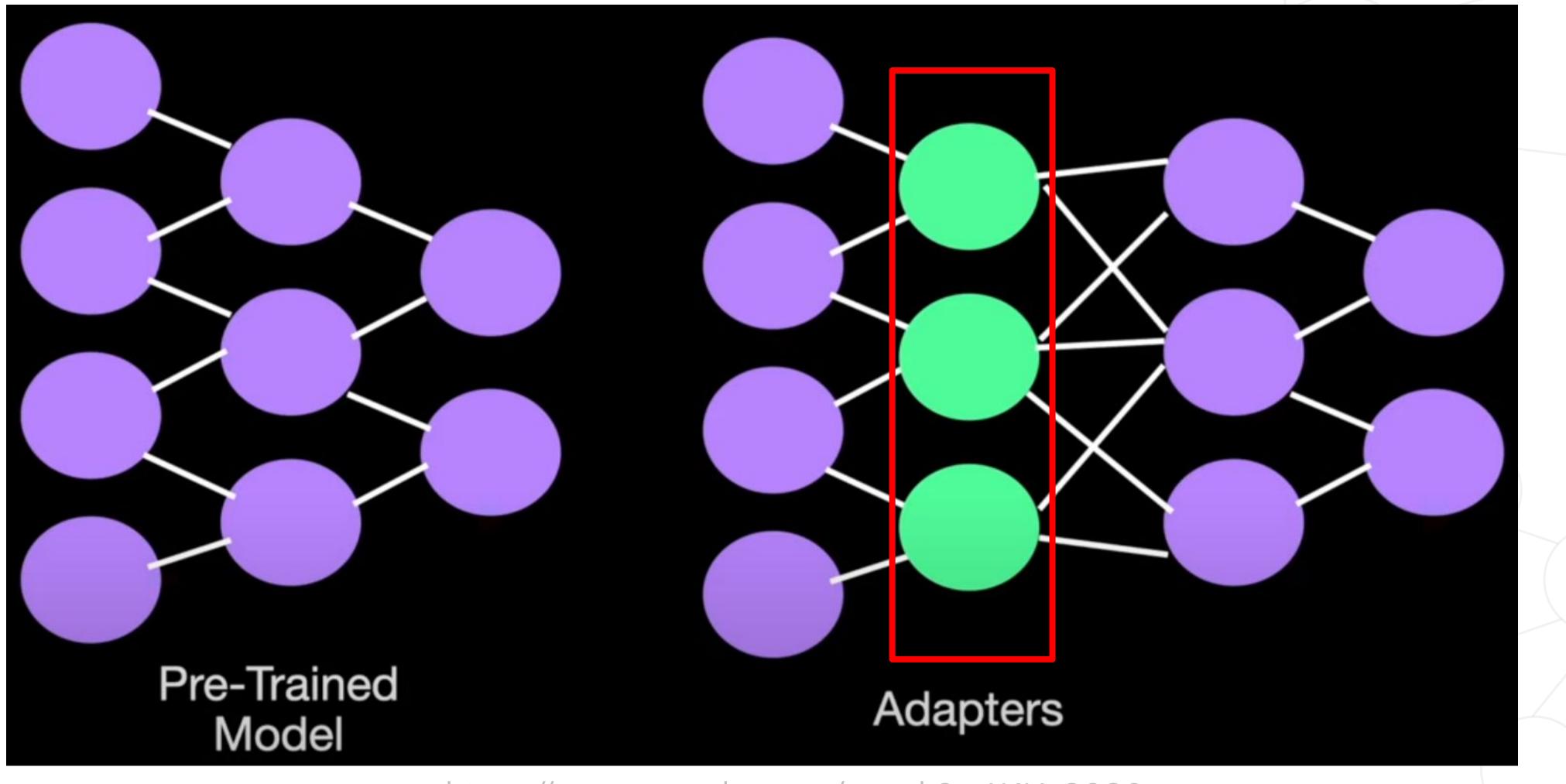
Generation of Synthetic Network Trace



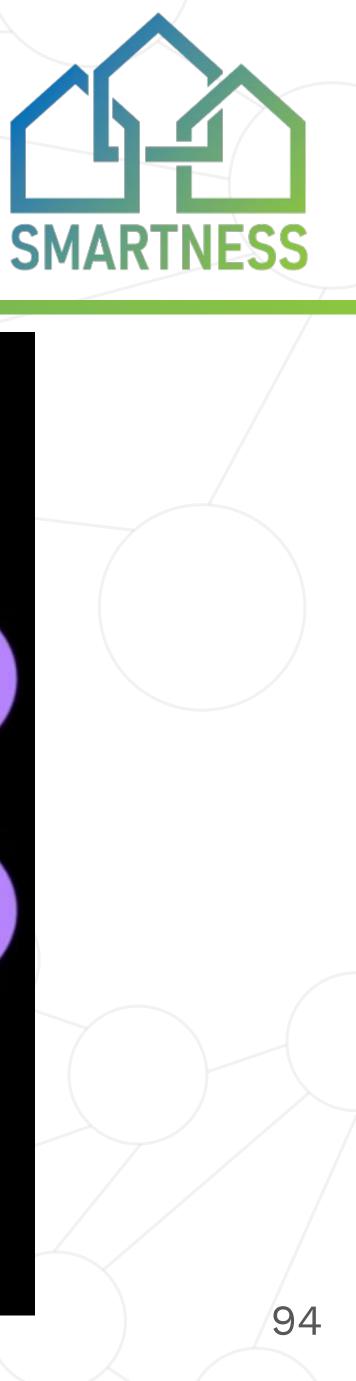


source: https://huggingface.co/blog/large-language-models

LoRa is an adapter Generation of Synthetic Network Trace



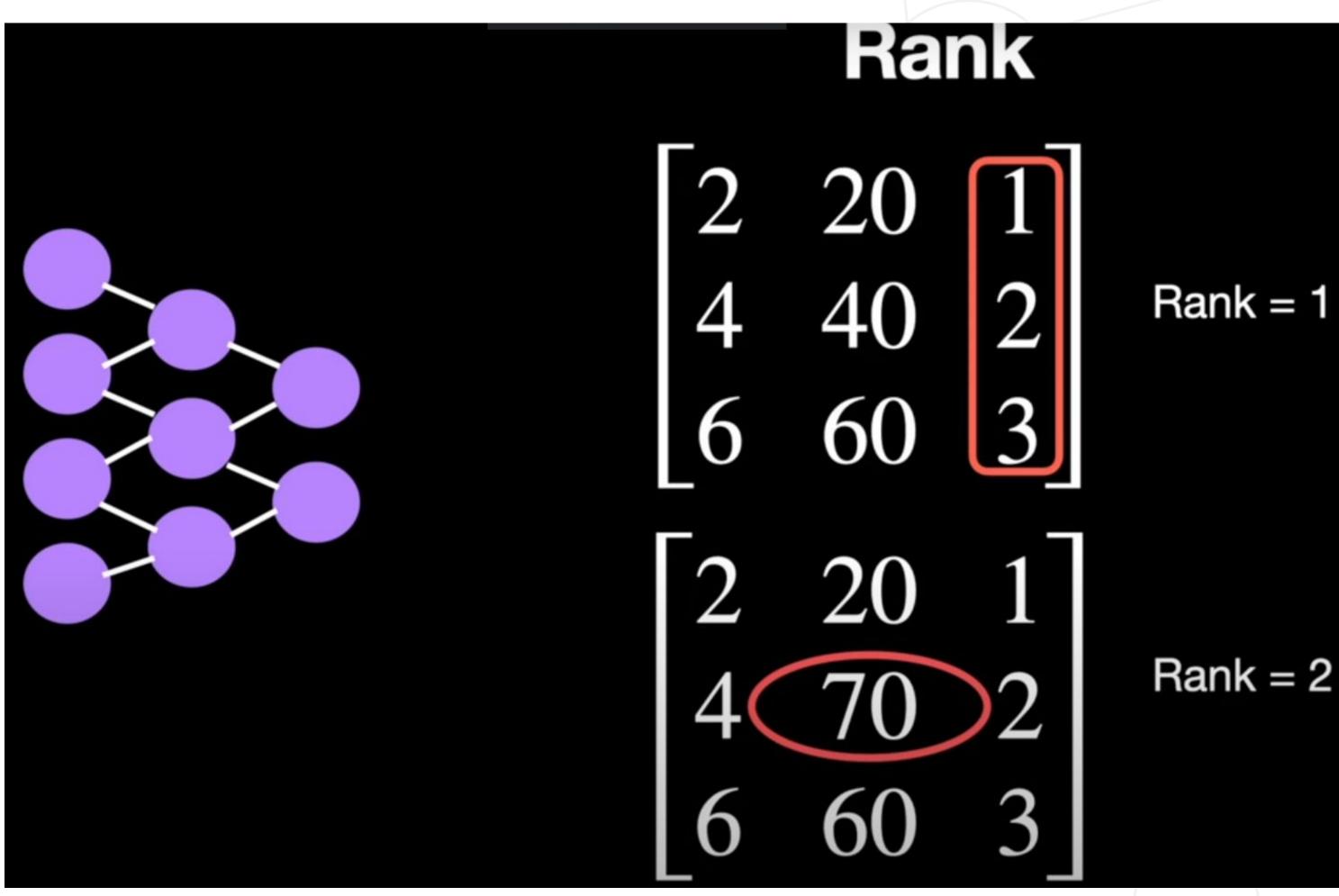
source: <u>https://www.youtube.com/watch?v=X4VvO3G6_vw</u>



LoRa leverages low-rank (of a matrix) Generation of Synthetic Network Trace

For instance, an 3x3 eye-matrix has rank 3, since all columns are independent

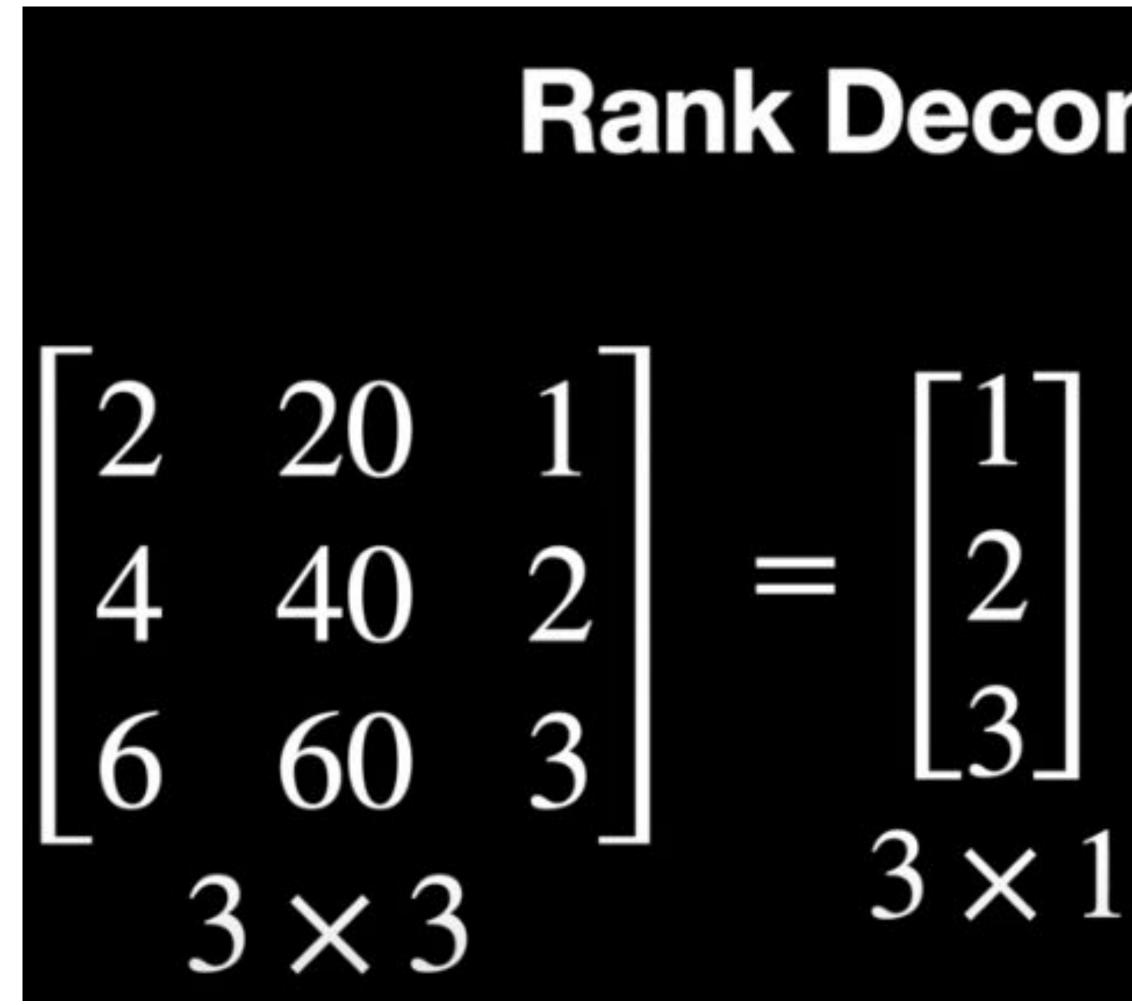
$$I_3 = egin{bmatrix} 1 & 0 & 0 \ 0 & 1 & 0 \ 0 & 0 & 1 \end{bmatrix}$$

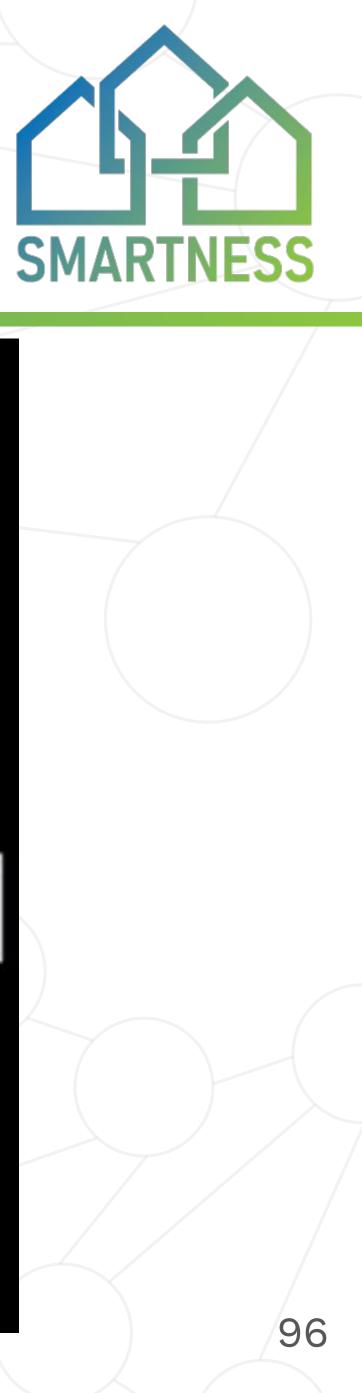




95

LoRa leverages low-rank (of a matrix) Generation of Synthetic Network Trace





Rank Decomposition

×[2 20 30] 3×1

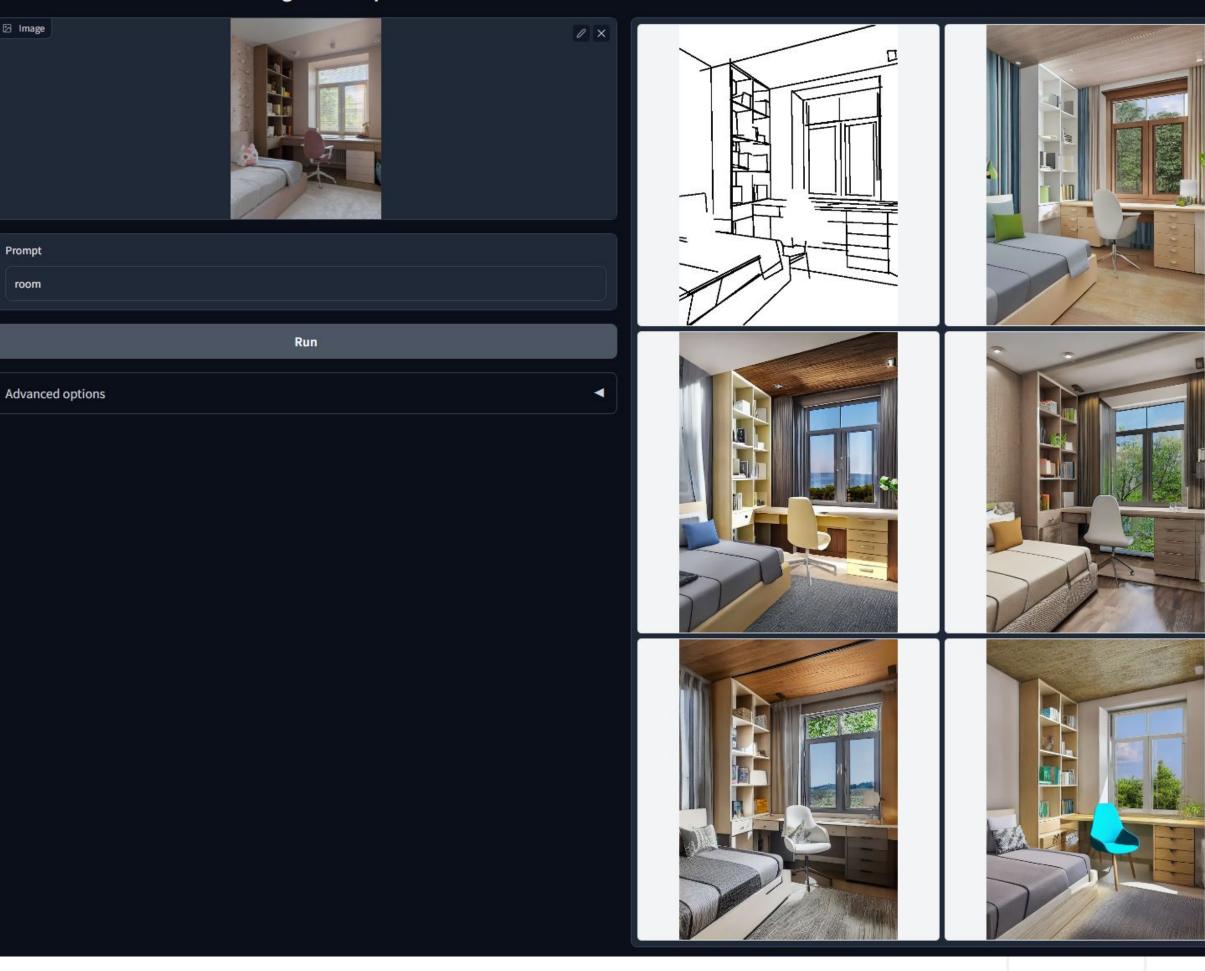
What is ControlNet?

Generation of Synthetic Network Trace

prompt: "room" ControlNet with M-LSD Lines



Control Stable Diffusion with Hough Line Maps



Generation of Synthetic Network Trace

7 Installing ControlNet extension

instatting controlliet extension
Diffusion checkpoint
-pruned-emaonly.safetensors [6ce0161689] 👻 🔄
img img2img Extras PNG Info Checkpoint Merger Train Settings Extensions
stalled Available Install from URL Backup/Restore
L for extension's git repository
ttps://github.com/Mikubill/sd-webui-controlnet
ecific branch name
eave empty for default main branch
cal directory name
eave empty for auto
Install





Generation of Synthetic Network Trace

8. Restarting WEB-UI and showing installed ControlNet extension

ble Diffusion	checkpoint							
v1-5-pruned-	-emaonly.safeten	sors [6ce01	61689] 👻					
txt2img	img2img	Extras	PNG Info	Checkpoint Merger	Train	Settings	Extensions	
Installed	Available	Instal	l from URL	Backup/Restore				
						Disable all ex	tensions	
A	Apply and resta	art UI		Check for updates		o none	🔵 extra 🔵 a	all
🔽 Exten	ision	URL			Branch	Version	Date	Update
🛃 LDSR		built-ir	1		None			
🛃 Lora		built-ir	1		None			
🛃 ScuNE	ET	built-ir	i .		None			3
🗹 Swinll	R	built-ir	1		None			
🗹 canva	s-zoom-and-pan	built-ir	1		None		а. Э	3
🗹 extra-	options-section	built-ir	1		None			
🛃 hyper	tile	built-ir	1		None			
🛃 mobil	e	built-ir	1		None			
🛃 prom	pt-bracket-check	er built-ir	1		None			
🗹 soft-in	npainting	built-ir	1		None			3
	bui-controlnet	https:/	github com/M	ikubill/sd-webui-controlnet	main	2091b6fb	2024-03-14 23:32:35	unknov



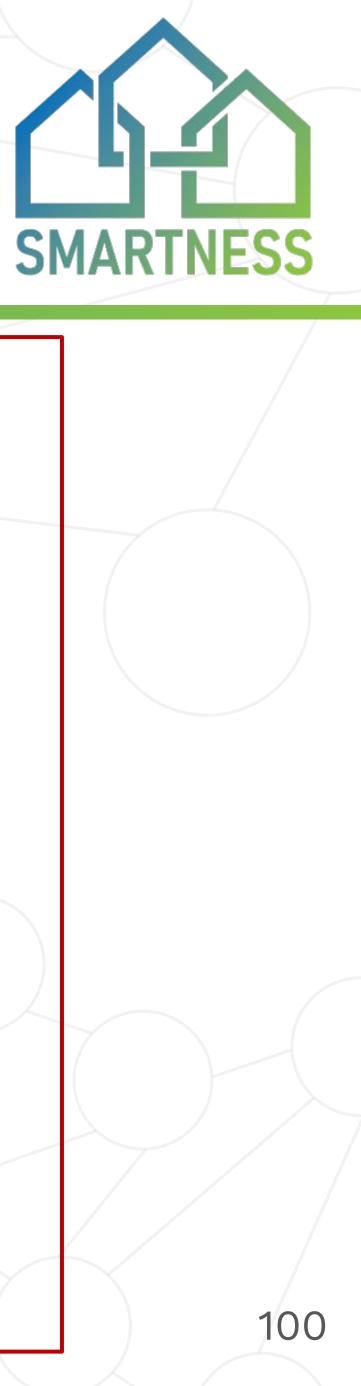
9. Showing the LoRA models

v1-5-prune	d-emaonly.safet	tensors [6ce0.	161689] 👻	1				
txt2img	img2img	Extras	PNG Info	Checkpoint Merger	Train	Settings	Extensions	
Prompt (Press Ctrl	+Enter to genera	ate, Alt+Enter	r to skip, Esc to i	interrupt)				
Negative p (Press Ctrl	prompt I+Enter to genera	ate, Alt+Enter	r to skip, Esc to i	interrupt)				
Generat	ion Textu	al Inversion	n Hyperne	etworks Checkpoint	s Lora			
			Т					
	- M/).	Ι	• NO •				
P	NO PREV); /TFH		: NO ; PREVEF	H			
P	: N(PREV); /TFH		: NO ; PREVEF	H			

Generation of Synthetic Network Trace

8. Restarting WEB-UI and showing installed ControlNet extension

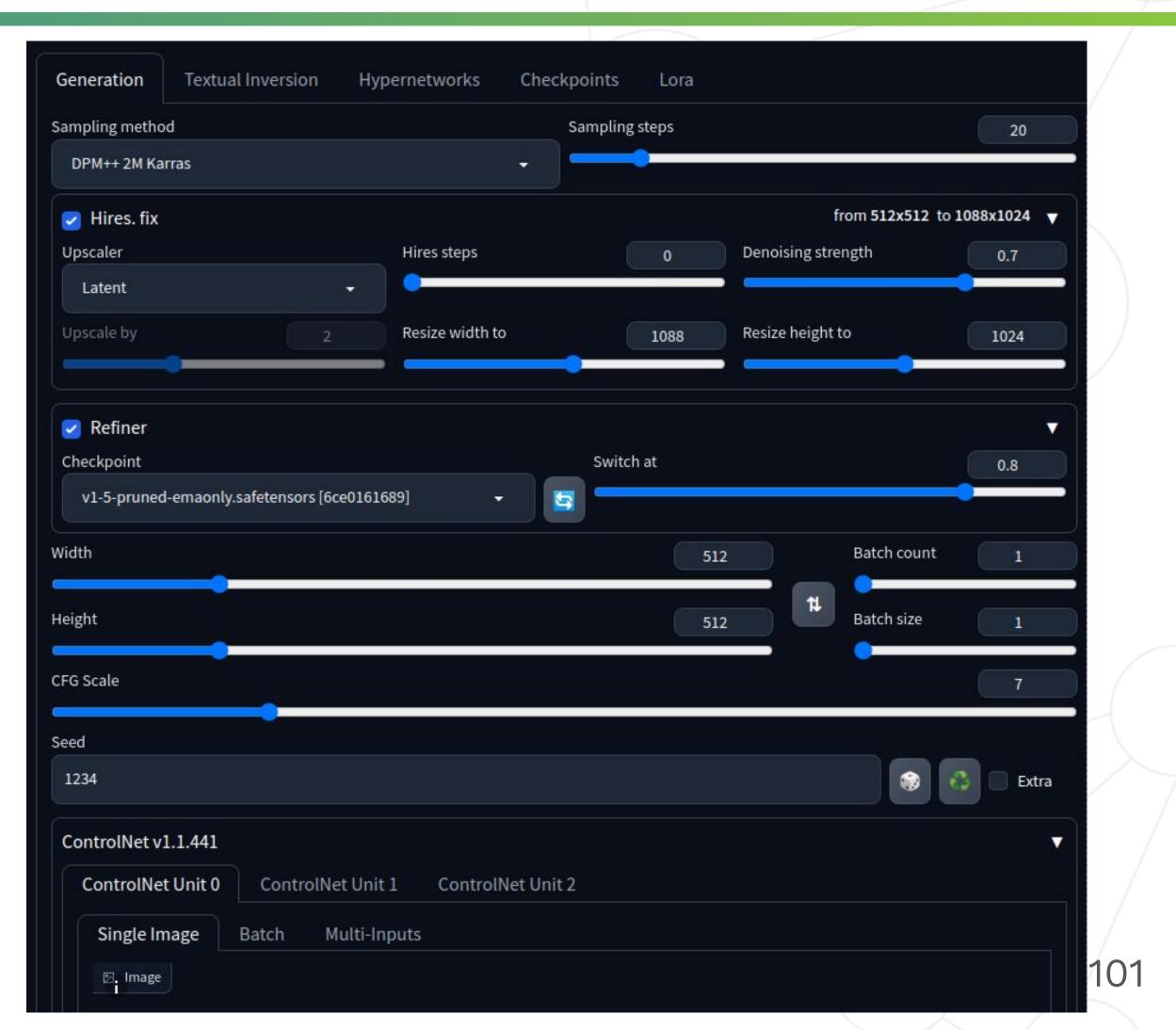
1-5-pruneo	d-emaonly.safetens	sors [6ce01	.61689] 👻					
xt2img	img2img	Extras	PNG Info	Checkpoint Merger	Train	Settings	Extensions	
Installed	Available	Instal	l from URL	Backup/Restore				
						Disable all e	tensions	
	Apply and resta	rt UI		Check for updates		💿 none	🔵 extra 🔵 a	att
🔽 Exte	nsion	URL			Branch	Version	Date	Updat
🛃 LDSF	ł	built-ir	built-in					
🗹 Lora		built-ir	1		None			
🗹 ScuN	IET	built-ir	built-in		None	21		3
🗹 Swin	IR	built-ir	built-in		None			
🗹 canv	as-zoom-and-pan	built-ir	built-in			2. 31		3
🗹 extra	-options-section	built-ir	built-in					
🗹 hype	ertile	built-ir	built-in					
🛃 mobile		built-ir	built-in		None			
	npt-bracket-checke	er built-ir	۱		None			
🗹 pron		built-ir			None	8		3
	inpainting	Duite-II			- Commencement	10	10	



9. Showing th	ne LoRA	\ mod	els	
Stable Diffusion checkpoint				
v1-5-pruned-emaonly.safetensors [6ce0161689]				
txt2img img2img Extras PNC	G Info Checkpoint Merger	Train Settings	Extensions	
Prompt (Press Ctrl+Enter to generate, Alt+Enter to skip	, Esc to interrupt)			
Negative prompt (Press Ctrl+Enter to generate, Alt+Enter to skip	Esc to interrupt)			
(Tress currenter to generate, Alt Enter to skip	Lise to interrupty			
Generation Textual Inversion H	lypernetworks Checkpoint	s Lora		
	110			
: NO :	(NO)			
PREVIFI	PREVIE	W		
Addams	lact			
Audams	last			$\langle \rangle$

Generation of Synthetic Network Trace

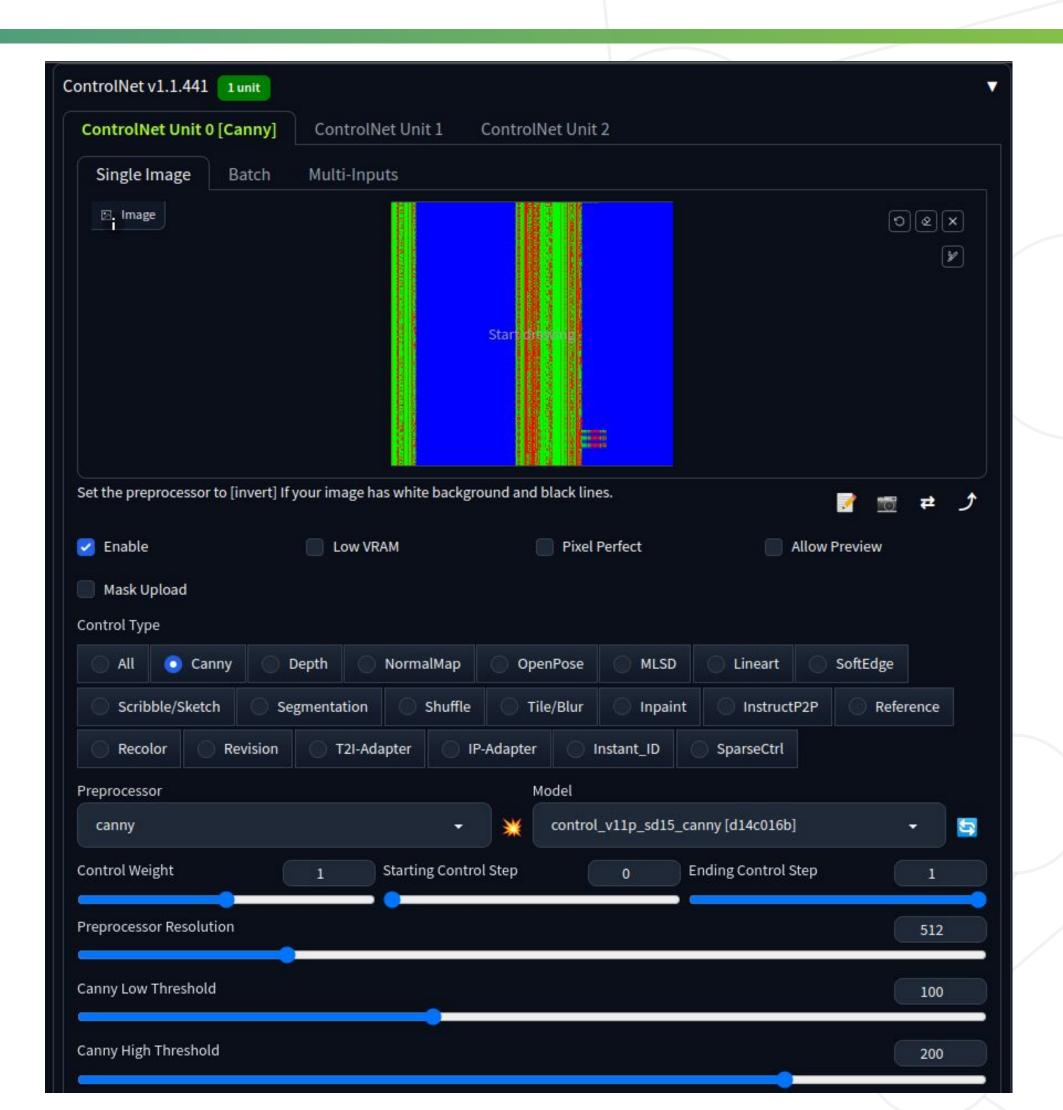
10. Upscaling the final image from 816x768 to 1088x1024

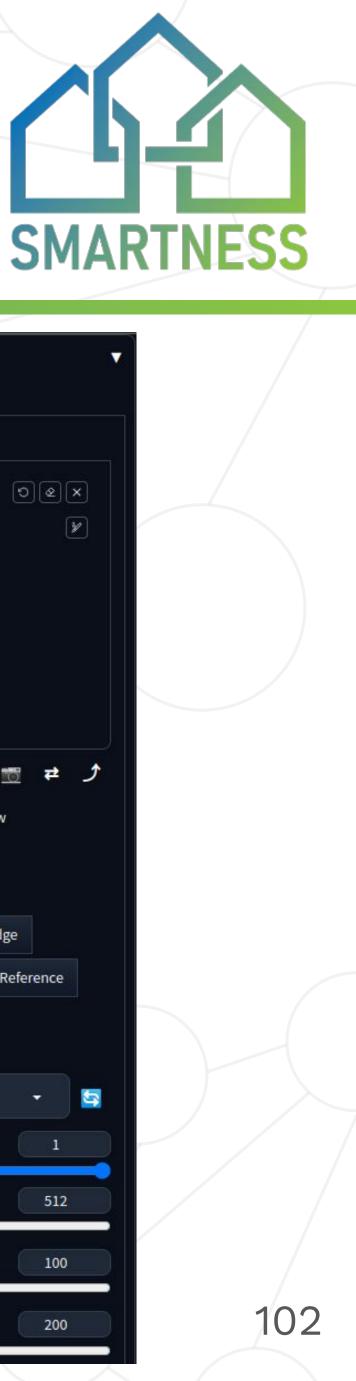




Generation of Synthetic Network Trace

11. Setting up ControlNet with Canny filter





Generation of Synthetic Network Trace

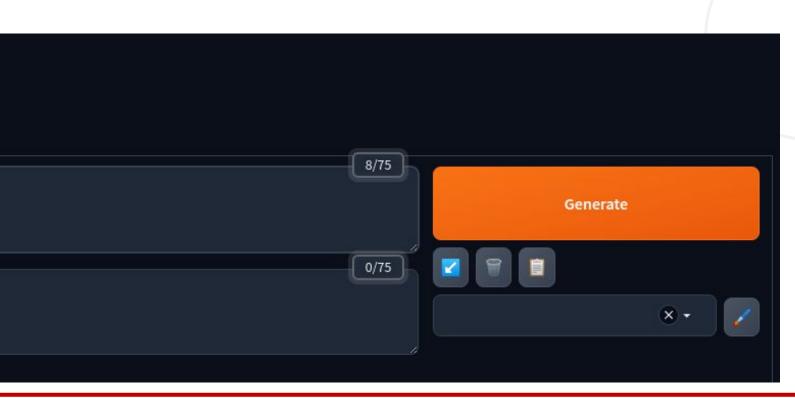
12. Setting the final prompt

				5			
t2img	img2img	Extras	PNG Info	Checkpoint Merger	Train	Settings	Extensions

13. Generating the final image

version-check commandline argument to disable this check. ntrolNet preprocessor location: /home/thiago/git_ariel/NetDiffusion_Generator/fine_tune/sd-webui-fork/stable-diffusion-webui/extensions/sd-webui-controlnet/annotator/downloads 24-03-26 12:34:47,722 - ControlNet - INFO - ControlNet v1.1.441 024-03-26 12:34:47,820 - ControlNet - INFO - ControlNet v1.1.441 pading weights [6ce0161689] from /home/thiago/git_ariel/NetDiffusion_Generator/fine_tune/sd-webui-fork/stable-diffusion-webui/models/Stable-diffusion/v1-5-pruned-emaonly.safetensors 024-03-26 12:34:48,014 - ControlNet - INFO - ControlNet UI callback registered. Inning on local URL: http://127.0.0.1:7860 o create a public link, set `share=True` in `launch()`. reating model from config: /home/thiago/git_ariel/NetDiffusion_Generator/fine_tune/sd-webui-fork/stable-diffusion-webui/configs/v1-inference.yaml plying attention optimization: Doggettx... done. odel loaded in 2.0s (load weights from disk: 0.6s, create model: 0.3s, apply weights to model: 1.0s). tartup time: 29.8s (prepare environment: 12.0s, import torch: 2.4s, import gradio: 0.4s, setup paths: 2.2s, other imports: 0.2s, load scripts: 0.8s, create ui: 0.5s, gradio launch: 11.2s). 2024-03-26 12:43:17,793 - ControlNet - INFO - unit_separate = False, style_align = False 2024-03-26 12:43:17,986 - ControlNet - INFO - Loading model: control_v11p_sd15_canny [d14c016b] 024-03-26 12:43:18,307 - ControlNet - INFO - Loaded state_dict from [/home/thiago/git_ariel/NetDiffusion_Generator/fine_tune/sd-webui-fork/stable-diffusion-webui/extensions/sd-webui-controlnet/models/co trol v11p sd15 canny.pth] 024-03-26 12:43:18,307 - ControlNet - INFO - controlnet default config 024-03-26 12:43:20,049 - ControlNet - INFO - ControlNet model control v11p sd15 canny [d14c016b](ControlModelType.ControlNet) loaded. 024-03-26 12:43:20,066 - ControlNet - INFO - Using preprocessor: canny 024-03-26 12:43:20,066 - ControlNet - INFO - preprocessor resolution = 512 24-03-26 12:43:20,214 - ControlNet - INFO - ControlNet Hooked - Time = 2.4237382411956787 20/20 [00:03<00:00, 6.33it/s 13/20 [00:14<00:07, 1.10s/it 55% otal progress: 82%|







Generation of Synthetic Network Trace

12. Setting the final prompt

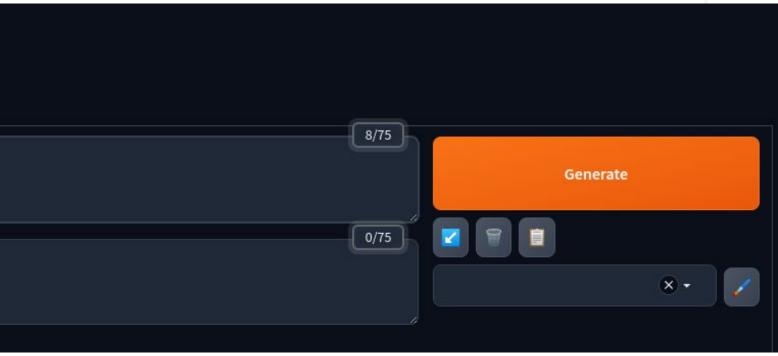
table Diffusio	n checkpoint			_				
v1-5-pruned	l-emaonly.safet	ensors [6ce0	161689] 👻					
txt2img	img2img	Extras	PNG Info	Checkpoint Merger	Train	Settings	Extensions	
pixelated n	etwork data, ty	pe-0 <lora:ad< th=""><th>ddams:1></th><th></th><th></th><th></th><th></th><th></th></lora:ad<>	ddams:1>					
Negative p								
(Press Ctrl-	Enter to genera	nte, Alt+Enter	to skip, Esc to ii	nterrupt)				

13. Generating the final image

version-check commandline argument to disable this check.

```
ntrolNet preprocessor location: /home/thiago/git_ariel/NetDiffusion_Generator/fine_tune/sd-webui-fork/stable-diffusion-webui/extensions/sd-webui-controlnet/annotator/downloads
 24-03-26 12:34:47,722 - ControlNet - INFO - ControlNet v1.1.441
024-03-26 12:34:47,820 - ControlNet - INFO - ControlNet v1.1.441
oading weights [6ce0161689] from /home/thiago/git_ariel/NetDiffusion_Generator/fine_tune/sd-webui-fork/stable-diffusion-webui/models/Stable-diffusion/v1-5-pruned-emaonly.safetensors
024-03-26 12:34:48,014 - ControlNet - INFO - ControlNet UI callback registered.
unning on local URL: http://127.0.0.1:7860
 o create a public link, set `share=True` in `launch()`.
reating model from config: /home/thiago/git_ariel/NetDiffusion_Generator/fine_tune/sd-webui-fork/stable-diffusion-webui/configs/v1-inference.yaml
 oplying attention optimization: Doggettx... done.
odel loaded in 2.0s (load weights from disk: 0.6s, create model: 0.3s, apply weights to model: 1.0s).
tartup time: 29.8s (prepare environment: 12.0s, import torch: 2.4s, import gradio: 0.4s, setup paths: 2.2s, other imports: 0.2s, load scripts: 0.8s, create ui: 0.5s, gradio launch: 11.2s).
                                2024-03-26 12:43:17,793 - ControlNet - INFO - unit_separate = False, style_align = False
024-03-26 12:43:17,986 - ControlNet - INFO - Loading model: control_v11p_sd15_canny [d14c016b]
024-03-26 12:43:18,307 - ControlNet - INFO - Loaded state dict from [/home/thiago/git_ariel/NetDiffusion_Generator/fine_tune/sd-webui-fork/stable-diffusion-webui/extensions/sd-webui-controlnet/models/co
trol v11p sd15 canny.pth]
024-03-26 12:43:18,307 - ControlNet - INFO - controlnet default config
2024-03-26 12:43:20,049 - ControlNet - INFO - ControlNet model control v11p sd15 canny [d14c016b](ControlModelType.ControlNet) loaded.
2024-03-26 12:43:20,066 - ControlNet - INFO - Using preprocessor: canny
024-03-26 12:43:20,066 - ControlNet - INFO - preprocessor resolution = 512
024-03-26 12:43:20,214 - ControlNet - INFO - ControlNet Hooked - Time = 2.4237382411956787
 55%
otal progress: 82%|
```





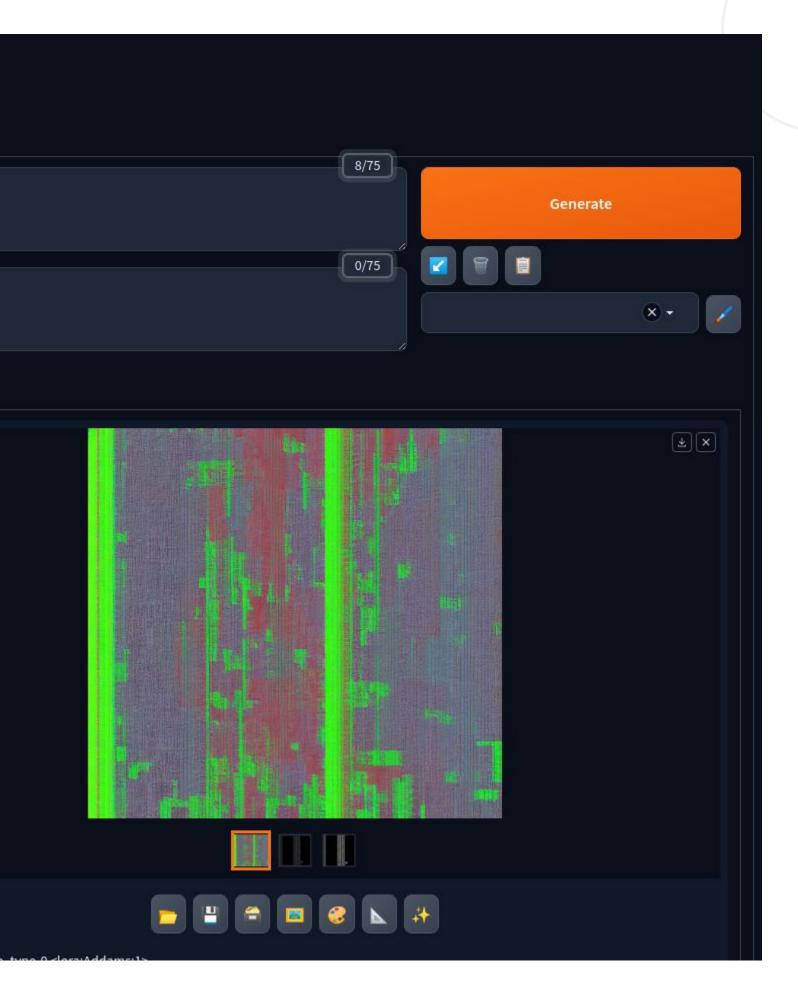
	20/20	[00:03<00:00,	6.33it/s]
	13/20	[00:14<00:07,	1.10s/it]
L	33/40	[00:20<00:07,	1.09s/it]

Generation of Synthetic Network Trace

14. Visualizing the final image

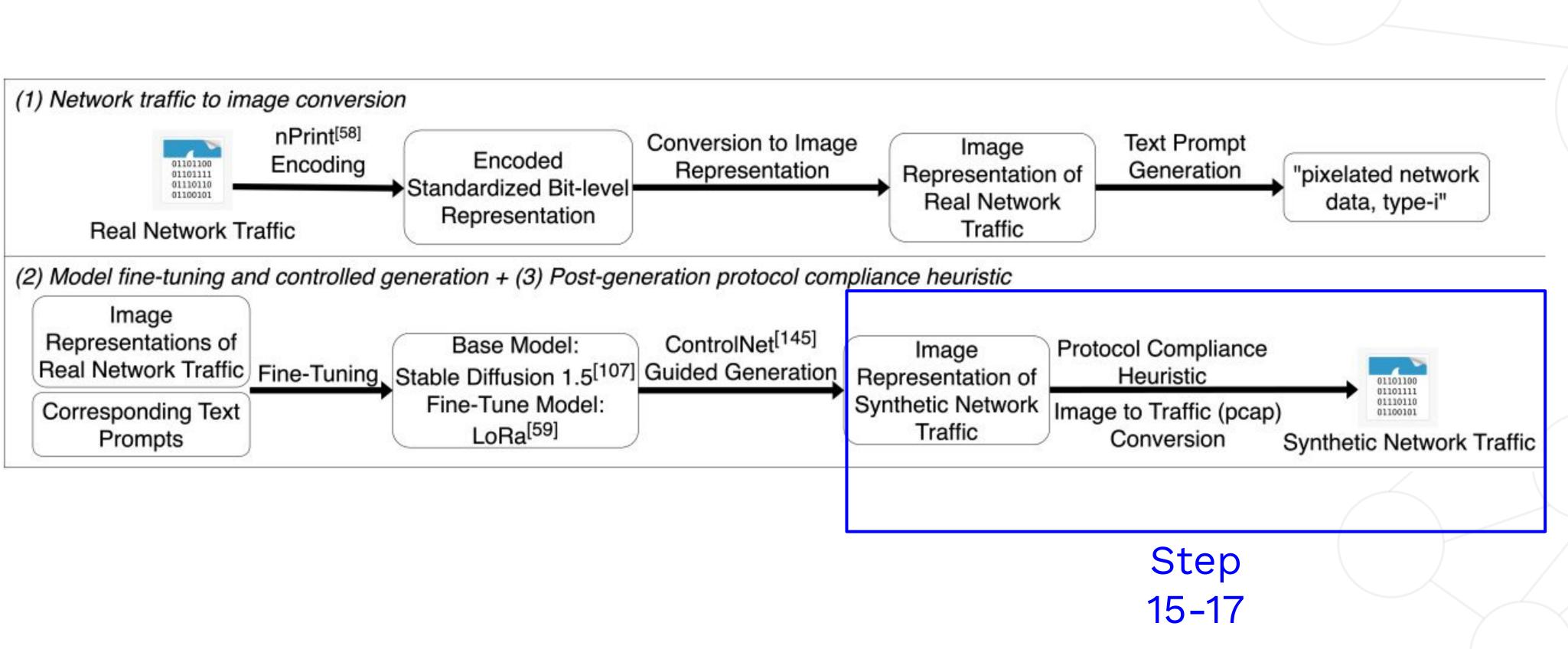
table Diffusion che v1-5-pruned-ema		ensors [6ce0]	.61689] 👻	5							
txt2img in	ng2img	Extras	PNG Info	Checkpoint N	lerger	Train	Settings	Extensions			
pixelated netwo	ork data, typ	pe-0 <lora:ad< td=""><td>dams:1></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></lora:ad<>	dams:1>								
Negative promp (Press Ctrl+Ente		te, Alt+Enter	to skip, Esc to	interrupt)							
Generation	Textua	al Inversion	Hypern	etworks Che	ckpoints	Lora					
Sampling meth	nod				Samplin	g steps			20		
DPM++ 2M K	arras			•		•					
🔽 Hires. fix	(from 512x512	2 to 1088x1024		
Upscaler			Hi	res steps		0	Denoisir	ng strength	0.7		
Latent											
Upscale by			2 Re	size width to		1088	Resize h	eight to	1024		
					-			-			
Refiner											
Checkpoint					Swit	ch at			0.8		
v1-5-prune	ed-emaonly.	.safetensors	[6ce0161689]	•	5						
Width					_		512	Batch cour	nt 1		
	-									=	
Height							512	Batch size	1		
CFG Scale									7		
Seed											

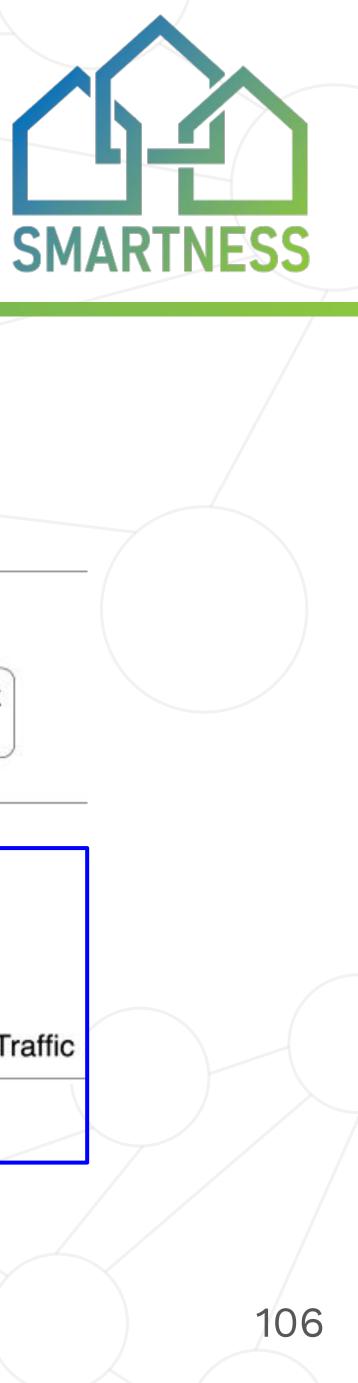




NetDiffusion: Workflow

Generation of Synthetic Network Trace





Generation of Synthetic Network Trace

15. Listing the final generated image

venv) (base) thiago@ifsuldeminas-Z390-M-GAMING:~/git_ariel/NetDiffusion_Generator/fine_tune/sd-webui-fork/stable-diffusion-webui/output/txt2img-images/2024-03-26\$ ls 00000-1234.png

venv) (base) thiago@ifsuldeminas-Z390-M-GAMING:~/git_ariel/NetDiffusion_Generator/fine_tune/sd-webui-fork/stable-diffusion-webui/output/txt2img-images/2024-03-26\$

16. Post-processing the final image

```
(base) thiago@ifsuldeminas-Z390-M-GAMING:~/git_ariel/NetDiffusion_Generator/post-generation$ python3 color_processor.py && python3 img_to_nprint.py && python3 mass_reconstruction.py
1024
088
024
024
 ./data/generated_nprint/00000-1234.nprint
 /data/replayable_generated_pcaps/00000-1234.pcap
 nome/thiago/git_ariel/NetDiffusion_Generator/post-generation/reconstruction.py:74: FutureWarning: Setting an item of incompatible dtype is deprecated and will raise an error in a future vers
 Value '199.180.150.116' has dtype incompatible with int64, please explicitly cast to a compatible dtype first.
 generated_nprint.at[idx, 'src_ip'] = implementing src ip
 ARNING: Inconsistent linktypes detected! The resulting file might contain invalid packets.
ARNING: Inconsistent linktypes detected! The resulting file might contain invalid packets.
ARNING: more Inconsistent linktypes detected! The resulting file might contain invalid packets.
 /data/generated_nprint/00008-1234.nprint
 /data/replayable_generated_pcaps/00008-1234.pcap
 home/thiago/git_ariel/NetDiffusion_Generator/post-generation/reconstruction.py:74: FutureWarning: Setting an item of incompatible dtype is deprecated and will raise an error in a future vers
 Value '24.43.115.154' has dtype incompatible with int64, please explicitly cast to a compatible dtype first.
 generated nprint.at[idx, 'src ip'] = implementing src ip
ARNING: Inconsistent linktypes detected! The resulting file might contain invalid packets.
 ARNING: Inconsistent linktypes detected! The resulting file might contain invalid packets.
ARNING: more Inconsistent linktypes detected! The resulting file might contain invalid packets.
 /data/generated nprint/netflix 5.nprint
 /data/replayable_generated_pcaps/netflix_5.pcap
 home/thiago/git_ariel/NetDiffusion_Generator/post-generation/reconstruction.py:74: FutureWarning: Setting an item of incompatible dtype is deprecated and will raise an error in a future vers
 Value '27.14.20.98' has dtype incompatible with int64, please explicitly cast to a compatible dtype first.
 generated_nprint.at[idx, 'src_ip'] = implementing_src_ip
ARNING: Inconsistent linktypes detected! The resulting file might contain invalid packets.
ARNING: Inconsistent linktypes detected! The resulting file might contain invalid packets.
 ARNING: more Inconsistent linktypes detected! The resulting file might contain invalid packets.
```



Generation of Synthetic Network Trace

15. Listing the final generated image

venv) (base) thiago@ifsuldeminas-Z390-M-GAMING:~/git_ariel/NetDiffusion_Generator/fine_tune/sd-webui-fork/stable-diffusion-webui/output/txt2img-images/2024-03-26\$ ls 00000-1234.png

venv) (base) thiago@ifsuldeminas-Z390-M-GAMING:~/git_ariel/NetDiffusion_Generator/fine_tune/sd-webui-fork/stable-diffusion-webui/output/txt2img-images/2024-03-26\$

16. Post-processing the final image

```
(venv) (base) thiago@ifsuldeminas-Z390-M-GAMING:~/git_ariel/NetDiffusion_Generator/post-generation$ python3 color processor.py && python3 img to nprint.py && python3 mass reconstruction.py
1088
1024
1088
1024
088
1024
 ./data/generated_nprint/00000-1234.nprint
 /data/replayable_generated_pcaps/00000-1234.pcap
 home/thiago/git_ariel/NetDiffusion_Generator/post-generation/reconstruction.py:74: FutureWarning: Setting an item of incompatible dtype is deprecated and will raise an error in a future vers
 . Value '199.180.150.116' has dtype incompatible with int64, please explicitly cast to a compatible dtype first.
 generated_nprint.at[idx, 'src_ip'] = implementing src ip
 ARNING: Inconsistent linktypes detected! The resulting file might contain invalid packets.
 ARNING: Inconsistent linktypes detected! The resulting file might contain invalid packets.
MARNING: more Inconsistent linktypes detected! The resulting file might contain invalid packets.
 /data/generated_nprint/00008-1234.nprint
 /data/replayable_generated_pcaps/00008-1234.pcap
 home/thiago/git_ariel/NetDiffusion_Generator/post-generation/reconstruction.py:74: FutureWarning: Setting an item of incompatible dtype is deprecated and will raise an error in a future ver
  Value '24.43.115.154' has dtype incompatible with int64, please explicitly cast to a compatible dtype first.
 generated_nprint.at[idx, 'src_ip'] = implementing_src_ip
 ARNING: Inconsistent linktypes detected! The resulting file might contain invalid packets.
 ARNING: Inconsistent linktypes detected! The resulting file might contain invalid packets.
 ARNING: more Inconsistent linktypes detected! The resulting file might contain invalid packets.
  /data/generated_nprint/netflix_5.nprint
 /data/replayable_generated_pcaps/netflix_5.pcap
 home/thiago/git ariel/NetDiffusion Generator/post-generation/reconstruction.py:74: FutureWarning: Setting an item of incompatible dtype is deprecated and will raise an error in a future vers
  Value '27.14.20.98' has dtype incompatible with int64, please explicitly cast to a compatible dtype first.
  generated_nprint.at[idx, 'src_ip'] = implementing_src_ip
ARNING: Inconsistent linktypes detected! The resulting file might contain invalid packets.
ARNING: Inconsistent linktypes detected! The resulting file might contain invalid packets.
 ARNING: more Inconsistent linktypes detected! The resulting file might contain invalid packets.
```





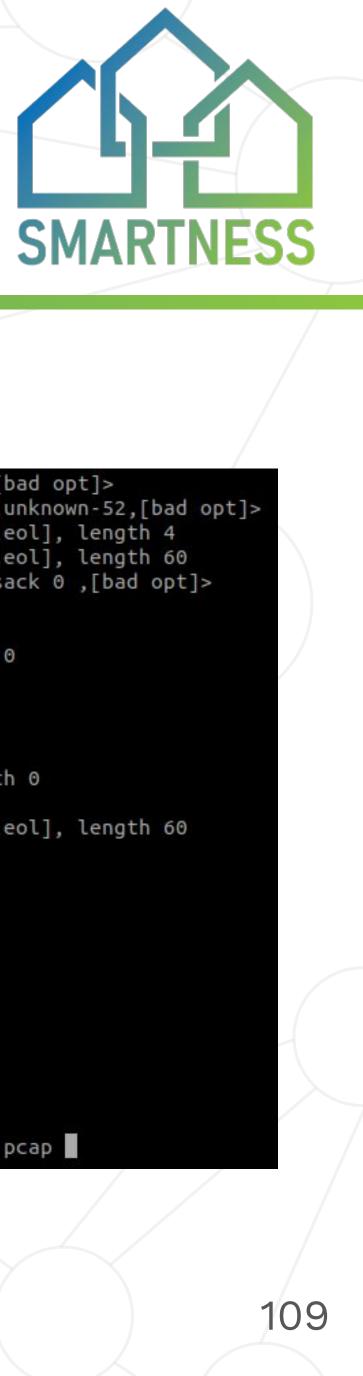


NetDiffusion: Main Steps Generation of Synthetic Network Trace

17. Testing the replayable PCAPs with tcpreplay

```
3:00:00.000000 IP 67.79.64.168.28917 > 199.180.150.116.44373: Flags [SR.], seq 2315281554, ack 2938939350, win 512, options [[bad opt]]
3:00:00.000000 IP 199.180.150.116.44373 > 67.79.64.168.28917: Flags [R.], seq 45, ack 0, win 0, options [sack 1 {2522978159:1996660334}, mss 416, eol], length 0
3:00:00.000000 IP 67.79.64.168.28917 > 199.180.150.116.44373: Flags [SR.], seq 2315281554, ack 2938939394, win 40, options [sack 1 {1362441987:1389648127},eol], length 0
3:00:00.000000 IP 67.79.64.168.28917 > 199.180.150.116.44373: Flags [.], ack 1, win 0, options [sack 1 {1899312896:1423268355},[bad opt]>
3:00:00.000000 IP 199.180.150.116.44373 > 67.79.64.168.28917: Flags [P.], seq 1:61, ack 0, win 640, options [sack 1 {2522978671:2013567724},eol], length 60
3:00:00.000000 IP 67.79.64.168.28917 > 199.180.150.116.44373: Flags [R.U], seq 0:64, ack 1, win 0, urg 0, options [[bad opt]
3:00:00.000000 IP 67.79.64.168.28917 > 199.180.150.116.44373: Flags [SRP.UEW], seq 2315281618:2315281658, ack 2938939394, win 256, urg 0, options [[bad opt]
3:00:00.000000 IP 199.180.150.116.44373 > 67.79.64.168.28917: Flags [SRP.], seq 2938939454, ack 2315281618, win 256, options [sack 1 {2520881712:2046925615},eol], length 0
 3:00:00.000000 IP 67.79.64.168.28917 > 199.180.150.116.44373: Flags [SR.], seq 2315281658, ack 2938939454, win 256, options [[bad opt]]
 3:00:00.000000 IP 199.180.150.116.44373 > 67.79.64.168.28917: Flags [SRP.], seq 2938939454:2938939514, ack 2315281658, win 120, options [sack 1 {2519832843:2013305607},eol], length 60
 User interrupt...
endpacket abort
Actual: 2014 packets (246652 bytes) sent in 0.317695 seconds
Rated: 776379.8 Bps, 6.21 Mbps, 6339.41 pps
lows: 2 flows, 6.29 fps, 4028 flow packets, 0 non-flow
Statistics for network device: eno1
       Successful packets:
                                  2013
       Failed packets:
       Truncated packets:
       Retried packets (ENOBUFS): 0
       Retried packets (EAGAIN): 0
     (base) thiago@ifsuldeminas-Z390-M-GAMING:~/git_ariel/NetDiffusion_Generator/data/replayable_generated_pcaps$ ^C
venv)
(venv) (base) thiago@ifsuldeminas-Z390-M-GAMING:~/git_ariel/NetDiffusion_Generator/data/replayable_generated_pcaps$ sudo tcpreplay --loop=0 --verbose -i eno1 00000-1234.pcap
```





3:00:00.000000 IP 67.79.64.168.28917 > 199.180.150.116.44373: Flags [R.], seq 61:113, ack 0, win 608, options [sack 1 {1358143278:1389648685}, unknown-36, sackOK[len 3], [bad opt]> 3:00:00.000000 IP 67.79.64.168.28917 > 199.180.150.116.44373: Flags [SR.], seq 2315281366:2315281430, ack 2938939350, win 1003, options [sack 1 {1358255916:1423268654}, unknown-52, [bad opt]> 3:00:00.000000 IP 67.79.64.168.28917 > 199.180.150.116.44373: Flags [SRP.], seq 2315281430:2315281434, ack 2938939350, win 530, options [sack 1 {1358180911:1389647915},eol], length 4 3:00:00.000000 IP 67.79.64.168.28917 > 199.180.150.116.44373: Flags [SRP.], seq 2315281434:2315281494, ack 2938939350, win 547, options [sack 1 {1899314735:1389648171},eol], length 60 3:00:00.000000 IP 67.79.64.168.28917 > 199.180.150.116.44373: Flags [SR.], seq 2315281494:2315281554, ack 2938939350, win 768, options [sack 1 {1358255660:1372871470}, sack 0 ,[bad opt]>



Conclusions and future perspectives

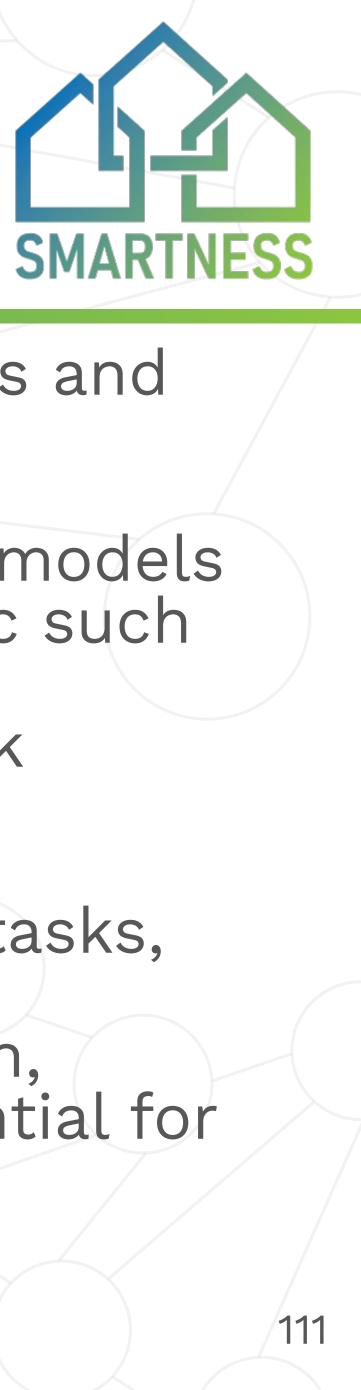




110

Conclusions and future perspectives

- Role of Generative Als: Simulate complex network environments and generate high-fidelity synthetic data, enhancing training for RL algorithms and network management.
- **Evolution and Application:** Development from basic generative models to advanced GANs capable of producing realistic network traffic such as PCAP files.
- **Practical Use:** Generating synthetic time series data for network telemetry and training ML models, particularly valuable in privacy-sensitive applications.
- Future Trends: Integration of GANs with network management tasks, promising innovative solutions for dynamic, complex systems. • **Research Opportunities:** Challenges in synthetic data generation, network simulation, and AI integration suggest significant potential for
- advancing network systems.



Thanks!

• Thiago Caproni Tavares

- o <u>thiago.caproni@ifsuldeminas.edu.br</u>
- Ariel Góes de Castro
- o <u>a272319@dac.unicamp.br</u> Leandro C. de Almeida
 - o leandro.almeida@ifpb.edu.br
- Washington Rodrigo Dias da Silva o <u>washingtonrds@estudante.ufscar.br</u>
- Christian Esteve Rothenberg o <u>chesteve@unicamp.br</u>
- Luciano Verdi o verdi@ufscar.br



