The 38th Annual AAAI Conference on Artificial Intelligence
FEBRUARY 20-27, 2024 | VANCOUVER, CANADA
VANCOUVER CONVENTION CENTRE - WEST BUILDING

UNIVERSITY OF SURREY

DiffSED: Sound Event Detection with Denoising Diffusion

Swapnil Bhosale[1]*, Sauradip Nag[1]*, Diptesh Kanojia[1] , Jiankang Deng[2] , Xiatian Zhu[1]

[1]University of Surrey, UK [2]Imperial College London, UK

Sound Event Detection (SED)

Frame-level SED

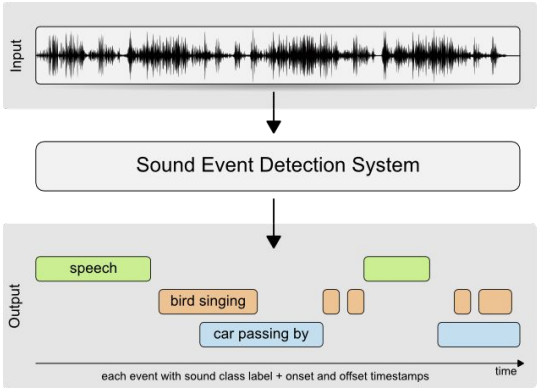
Classify each audio frame/segment into event classes and aggregate consecutive frame-level predictions.

- heavily manually designed with plenty of heuristics.
- not easily scalable, converted into sequence multilabel classification task.

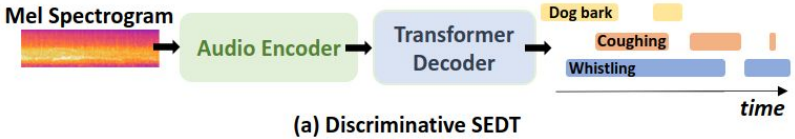
Event-level SED

Directly learn the correlation between frames.

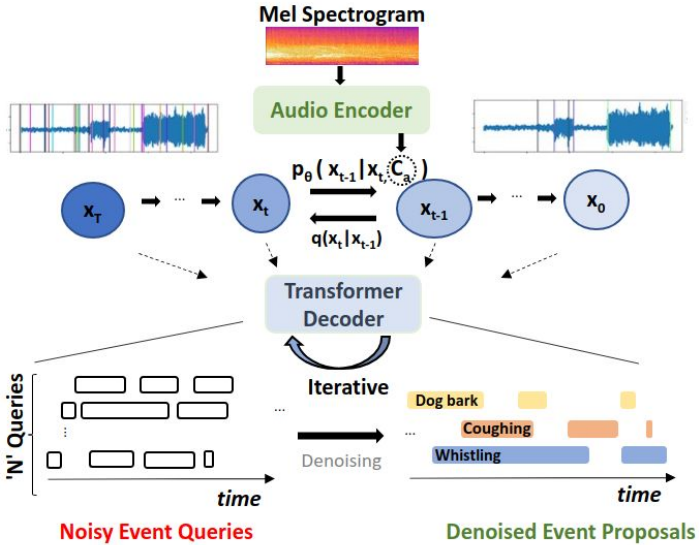
- eliminates the mundane post-processing.
- more generalizable.



Proposal



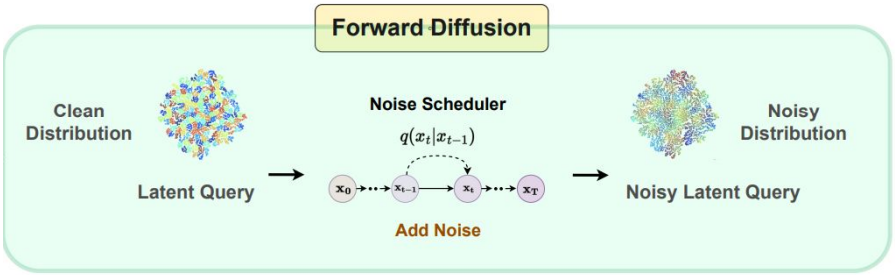
(a) Discriminative SEDT



(b) Generative SEDT (ours)

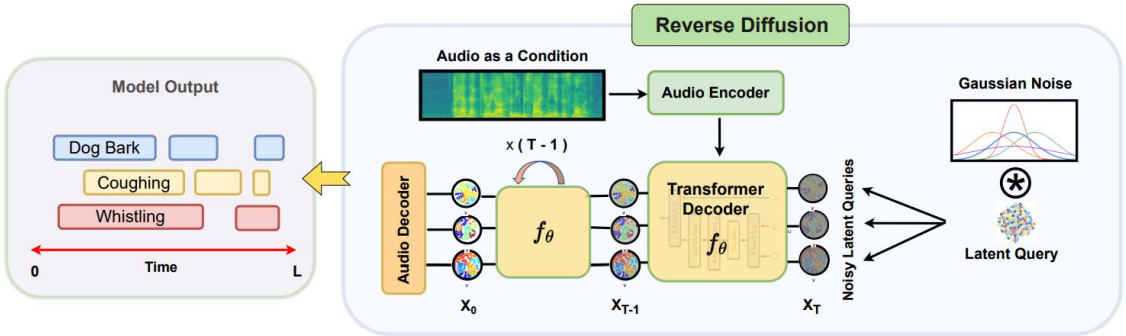
- Reformulate SED as a generative denoising process in an elegant transformer decoder framework.
- Generative adaptation using a noise-to-queries strategy (i.e. Diffusion-based), with *evolutionary enhancement of queries* and *faster convergence*.

Methodology - A Conditional Denoising Framework



Forward Diffusion: Gaussian noises are added to the event latents iteratively to obtain noisy latents X_T

Reverse Diffusion: Audio melspectrogram is passed as the condition along with random noisy latents sampled from the Gaussian distribution.
Noisy latents are passed as the query to the denoiser for denoising the event latents in an iterative fashion to obtain event proposals.



Training Algorithm

- Learnable event query embeddings, z_0 - retrieved from a lookup table that stores embeddings of a fixed dictionary of size N (gaussian initialized)
- Task : Train a neural network to predict (\progressively refine) z_0 from noisy proposals z_t, conditioned on the input audio (encoded).
- Output from the last denoising step (corresponding to each input event query) is projected into sigmoidal onset and offset timestamps and an event probability distribution using separate feedforward projection layers.

Algorithm 1 Training

```
def train_loss(audio, event_query):
"""
audio: [B, T, F]
event_queries: [B, N, D]
# B: batch_size
# N: number of event queries
"""
# Encode audio features
audio_feats = audio_encoder(audio)

# Signal scaling
event_queries = (event_queries * 2 - 1) * scale

# Corrupt event_queries
t = randint(0, T)              # time step
eps = normal(mean=0, std=1) # noise: [B, N, D]
event_queries_crpt = sqrt(    alpha_cumprod(t)) * event_queries +
 sqrt(1 - alpha_cumprod(t)) * eps

# Predict bounding boxes
pb_pred = detection_decoder(event_queries_crpt, audio_feats, t)

# Set prediction loss
loss = set_prediction_loss(pb_pred, gt_boxes)

return loss
```

Noise Corruption

With inherent query based design, two denoising strategies can be designed:

- Corrupting the event latents in the continuous space and passing it as queries.
- Corrupting discrete event proposals (i.e., groundtruth bounding boxes) and projecting it as queries.

Algorithm 2 Noise corruption

```
def add_noise():
"""
gt_boxes: [B, *, 2]
event_queries: [B, N, D]
B: batch_size
N: number of event queries
"""
if corrupt bounding_boxes: # Diff-SED-BB
    # Padding (repeat) bounding boxes
    pb = Pad(gt_boxes, N) #[B, N, 2]
    # Signal scaling
   pb = (pb * 2 - 1) * scale
   # Corrupt bounding boxes
   t = randint(0, T)            #time step
   eps = normal(mean=0, std=1) #noise: [B, N, 2]
   pb_crpt = sqrt(alpha_cumprod(t)) * pb + sqrt(1 - alpha_cumprod(t)) * eps
    event_queries_crpt = Project(pb_crpt)
    #[B, N, 2] -> [B, N, D]
 else: # DiffSED
   # Signal scaling
   event_queries = (event_queries * 2 - 1) * scale
   # Corrupt event_queries
   t = randint(0, T)            #time step
   eps = normal(mean=0, std=1) #noise: [B, N, D]
   event_queries_crpt = sqrt(alpha_cumprod(t)) * event_queries +
    sqrt(1 - alpha_cumprod(t)) * eps
return event_queries_crpt
```

Inference Algorithm

Denoising Sampling

Denoised event queries from the denoising decoder are decoded using two parallel heads,

- Event Classification head
- Event Localization head

Algorithm 3 Inference

```
def infer(audio, event_query):
"""
audio: [B, T, F]
event_queries: [B, N, D]
# B: batch_size
# N: number of event queries
"""
# Encode audio features
audio_feats = audio_encoder(audio)

# uniform sample step size
times = reversed(linespace(-1, T, steps))

# [(T-1, T-2), (T-2, T-3), ..., (1,0), (0,-1)]
time_pairs = list(zip(times[:-1], times[1:])

if corrupt bounding_boxes: # Diff-SED-BB
   # noisy boxes: [B, N, 2]
   pb_t = normal(mean=0, std=1)
   #Project [B, N, 2] into [B, N, D]
   event_queries_t = Project(pb_t)
else:
   # noisy event queries: [B, N, D]
   event_queries_t = normal(mean=0, std=1)

for t_now, t_next in zip(time_pairs):

  # Predict event_queries_0 from event_queries_t
  event_queries_pred = detection_decoder(event_queries_t, audio_feats, t_now)

  # Estimate event_queries_t at t_next
  event_queries_t = ddim_step(event_queries_t, event_queries_pred, t_now, t_next)

  # Event query renewal
  event_queries_t = box_renewal(event_queries_t)

#Project [B, N, D] into [B, N, 2], i.e. event query to event boundary
pb_pred = pb_projector(event_query_pred)
return pb_pred
```

Results

Wide range of sound classes, including human speech, animal sounds, environmental sounds, and music.

- URBAN-SED
  - onset, offset for each sound event
- EPIC-Sounds
  - 36k audio recordings

Table 1: Results on URBAN-SED (Test set)

Model	Event-based [%]			Segment-based[%]			Audio tagging[%]
	F1	P	R	F1	P	R	F1
CRNN-CWin [48]	36.75	–	–	65.74	–	–	74.19
Ctrans-CWin [48]	34.36	–	–	64.73	–	–	74.05
SEDT [65]	37.27	43.32	33.21	65.21	74.82	58.46	74.37
DiffSED (**Ours**)	**43.89**	**48.46**	**37.82**	**69.24**	**77.49**	**62.05**	**77.87**

Table 2: Results on EPIC-Sounds (Validation set)

Model	Top-1	Top-5	mCA	mAP	mAUC
ASF [25]	53.47	84.56	20.22	0.235	**0.879**
SSAST [18]	53.75	84.54	20.11	0.254	0.873
DiffSED (**Ours**)	**56.85**	**87.45**	**20.75**	**0.277**	0.861

Ablation

Table 3: Effect of the number of queries on the performance for URBAN-SED Test set. (AT: Audio Tagging performance)

	#Queries	Event-F1[%]	Segment-F1[%]	AT[%]
DiffSED-BB	10	31.43	58.85	68.87
	20	35.32	60.53	68.84
	30	**37.29**	**60.91**	**69.61**
	40	31.95	58.79	68.41
	50	31.81	57.89	68.31
DiffSED	10	40.78	68.41	77.22
	20	**41.42**	**68.73**	76.54
	30	41.3	68.21	**77.46**
	40	38.65	67.21	75.21
	50	36.28	64.22	72.77

Table 4: Effect of the number of denoising steps used while inference on the performance for URBAN-SED Test set. (AT: Audio Tagging performance)

	#steps	Event-F1[%]	Segment-F1[%]	AT[%]
DiffSED-BB	1	**39.78**	64.74	**72.92**
	5	38.27	**65.72**	71.88
	10	38.3	64.82	72.17
DiffSED	1	43.89	69.24	**77.87**
	5	**44.35**	**70.75**	77.07
	10	43.50	69.05	77.36

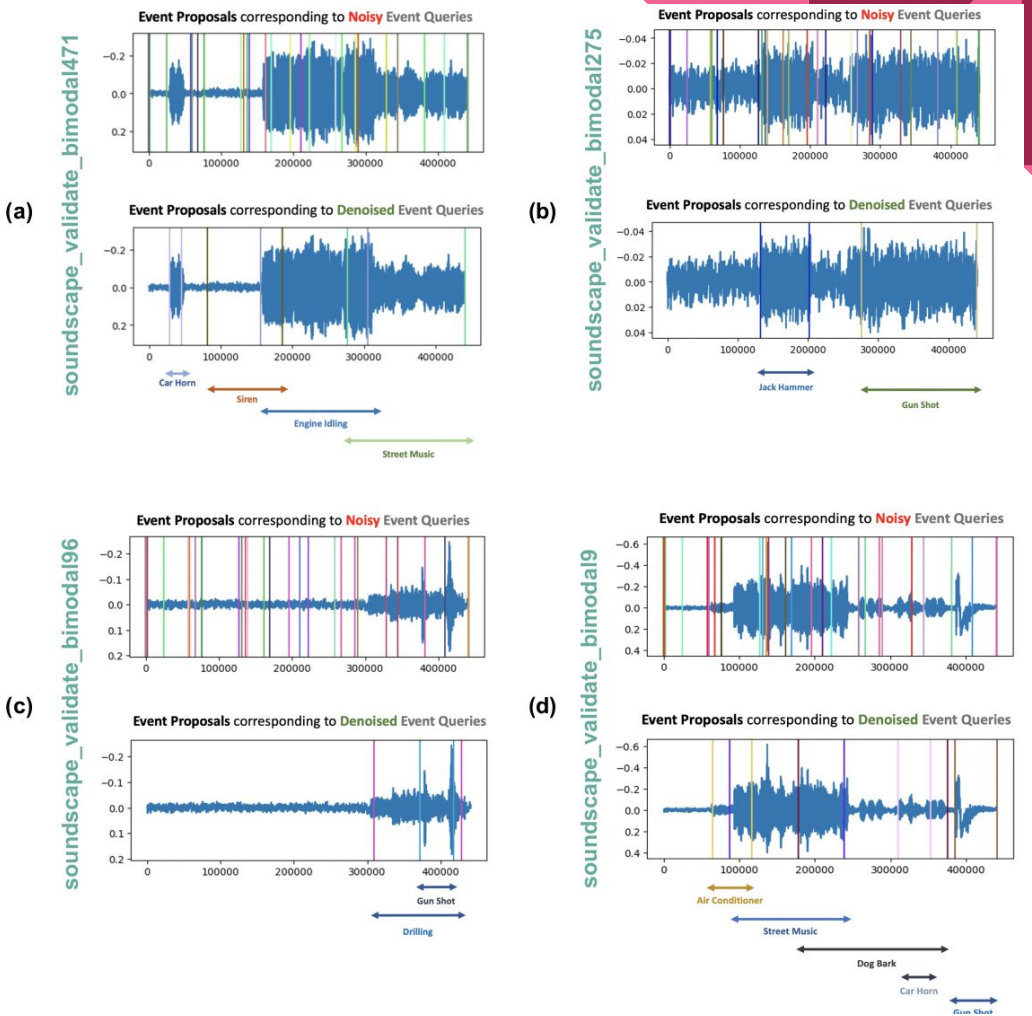
Table 5: Effect of scaling the noise factor on the performance for URBAN-SED Test set. (AT: Audio Tagging Performance)

	Noise scale	Event-F1[%]	Segment-F1[%]	AT[%]
DiffSED-BB	0.1	32.61	32.45	73.49
	0.2	35.91	35.73	**75.73**
	0.3	37.29	60.91	69.61
	0.4	**39.78**	**64.74**	72.92
	0.5	33.14	61.79	71.12
DiffSED	0.1	37.61	54.63	72.2
	0.2	39.65	58.17	73.89
	0.3	41.3	68.21	77.46
	0.4	**43.89**	**69.24**	**77.87**
	0.5	39.23	59.25	72.78

Table 6: Effect of changing the seed value for inducing noise during inference. Values inside (.) indicate deviation from the mean calculated over 3 runs.

	Runs	Event-F1[%]	Segment-F1[%]	AT[%]
DiffSED-BB	1	38.6(↑ 0.2)	64.32(↓ 0.09)	72.48(0.0)
	2	39.45(↓ 0.57)	64.15(↑ 0.07)	72.88(↓ 0.4)
	3	38.57(↑ 0.3)	64.21(↑ 0.01)	72.08(↑ 0.4)
	Avg	**38.87**	**64.22**	**72.48**
DiffSED	1	43.12(↓ 0.2)	68.38(↑ 0.5)	77.62(↓ 0.01)
	2	42.35(↑ 0.5)	68.97(↑ 0.01)	77.59(↑ 0.02)
	3	43.29(↓ 0.3)	69.54(↓ 0.5)	77.62(↓ 0.01)
	Avg	**42.92**	**68.96**	**77.61**

Visualizations



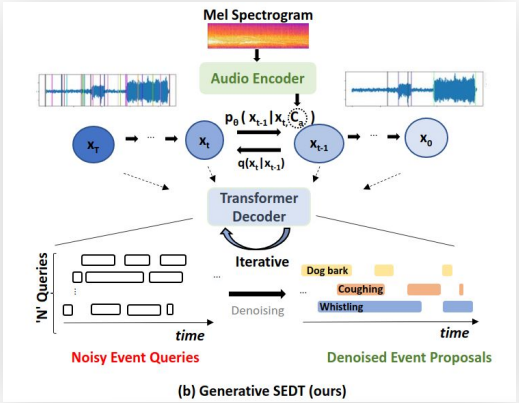
The 38th Annual AAAI Conference on Artificial Intelligence

DiffSED: Sound Event Detection with Denoising Diffusion

Swapnil Bhosale[1]*, Sauradip Nag[1]*, Diptesh Kanojia[1], Jiankang Deng[2], Xiatian Zhu[1]
[1]University of Surrey, UK [2]Imperial College London, UK

Highlights:

- Reformulate SED as a generative denoising process in an elegant transformer decoder framework.
- Generative adaptation using a noise-to-queries strategy (i.e. Diffusion-based), with evolutionary enhancement of queries and faster convergence (at least 40%).



Forward Diffusion: Gaussian noises are added to the event latents iteratively to obtain noisy latents X_T
Reverse Diffusion: Audio is passed as the condition along with random noisy latents.

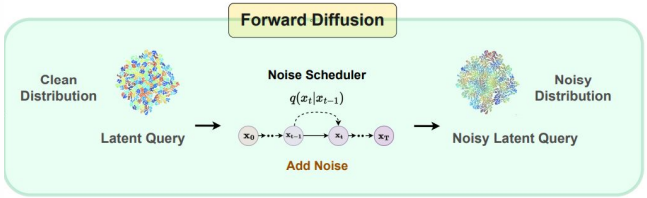
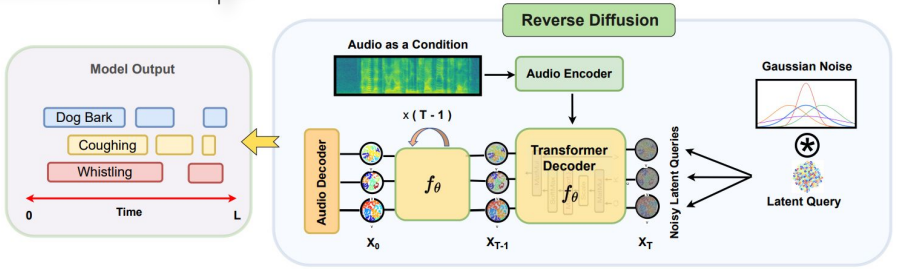


Table 1: Results on URBAN-SED (Test set)

Model	Event-based [%]			Segment-based[%]			Audio tagging[%]
	F1	P	R	F1	P	R	F1
CRNN-CWin (Miyazaki et al. 2020b)	36.75	–	–	65.74	–	–	74.19
Ctrans-CWin (Miyazaki et al. 2020b)	34.36	–	–	64.73	–	–	74.05
SEDT (Ye et al. 2021)	37.27	43.32	33.21	65.21	74.82	58.46	74.37
DiffSED (**Ours**)	**43.89**	**48.46**	**37.82**	**69.24**	**77.49**	**62.05**	**77.87**

Table 2: Results on EPIC-Sounds (Validation set)

Model	Top-1	Top-5	mCA	mAP	mAUC
ASF (Kazakos et al. 2021)	53.47	84.56	20.22	0.235	**0.879**
SSAST (Gong et al. 2022)	53.75	84.54	20.11	0.254	0.873
DiffSED (**Ours**)	**56.85**	**87.45**	**20.75**	**0.277**	0.861



Noisy latents are passed as the query to the denoiser for denoising the event latents in an iterative fashion to obtain event proposals.

Bhosale, S., Nag, S., Kanojia, D., Deng, J., & Zhu, X. (2023). DiffSED: Sound Event Detection with Denoising Diffusion. arXiv preprint arXiv:2308.07293. Accepted AAAI 2024