109-2 MMSEC: Final Project

R08945027 何明洋

Which layer make deep learning model vulnerable: embedding analysis of adversarial attack.

Code available at: https://github.com/Kaminyou/Adversarial-attack-embedding-analysis

1. Introduction

While deep learning models have been utilized to tackle several tasks in modern life, their vulnerability against adversarial examples is well-known [4], which can be easily deceived by these adversaries. The situation will be more dangerous if the model is leveraged to deal with critical tasks, for example, clinical diagnosis [1, 2, 3] or decision of self-driving car [5].

Several algorithms have been published to generate adversarial examples and succeed in deceiving deep learning models no matter in white-box or black-box scenario. Start from the one-step attempt such as FGSM [6] algorithm to the iterative-based methods like PGD [7] and MI-FGSM [8] algorithms, and recently, manifold strategies considering the generative models are also leveraged to generate adversarial examples with constraint other than L-inf that often utilized in the old algorithm.

Although the threat they imposed, the exact mechanism why a deep model is vulnerable to the adversarial examples is not fully studied and unknown yet. Hence, this study attempts to investigate and explicate the possible causes by analyzing the embedding of the adversarial examples from each layer of models to find out the most unprotected layer in whole model, which might finally cause the misclassification.

The ImageNette dataset is utilized in this study as the source to generate adversarial example through FGSM, PGD, MI-FGSM, and DeepFool [9] algorithms with the models of pretrained and fine-tuned ResNet-18, ResNet-50 [10], DenseNet-121 [11], and wide ResNet-50 v2 [12] considering both targeted and untargeted attacks. A cross-model attack experiment will be conducted to examine the transferability of the adversarial examples and the robustness of these models. Finally, the embedding from each layer will be projected via t-SNE [13] algorithm and visualized for investigation of vulnerable layer.

To the best of our knowledge, this is the first study to investigate the embedding from each layer against adversarial examples to find out the vulnerable layer and perform further analysis.

2. Method

2.1. Dataset

The ImageNette dataset [14] with classes of tench, English springer, cassette player, chain saw, church, French horn, garbage truck, gas pump, golf ball, and parachute, which is a subset of ImageNet that consists images of 1000 classes, is utilized in this study to simplify the problem. All the images will be resized to 256*256 pixel, cropped by the center to 224*224 pixel, and normalized with the means of 0.485, 0.456, 0.406 and standard deviation of 0.229, 0.224, 0.225 before input to a model.

2.2. Model

Four popular models are utilized in this studying, including ResNet-18, ResNet-34, DenseNet-121, and wide ResNet-50 v2. These models are initialized with the pretrained weighted obtained by the training with full ImageNet. The last fully-connected layer will be reset from the size of 1000 to the 10 that corresponding with the number of classes of ImageNette. The model is further fine-tuned by SGDM optimization algorithm with epochs of 10, batch size of 50, learning rate of 0.01, momentum of 0.9, and l2 regularization of 0. 00001.

2.3. Adversarial examples generation

After obtaining the models trained by ImageNette dataset, four algorithms are leveraged to generate adversarial examples:

2.3.1. FGSM algorithm

FGSM algorithm, a relative efficient attack technique, was proposed by Goodfellow, et al. This algorithm directly leverages the gradient of input x to find direction to perturb x and achieve the optimization problem. The formula is as below:

$$x' = \mathrm{x} + \varepsilon \cdot \mathrm{sign}(\nabla_x \mathcal{L}_\theta(x, y))$$

With this formula, an untargeted attack can be achieved. For the targeted attack, it is simply to adjust this formula as:

$$x' = \mathrm{x} - \varepsilon \cdot \mathrm{sign}(\nabla_x \mathcal{L}_\theta(x, y'))$$

This method perturbs x in infinity-norm space, which is also reasonable to human considering in the computer vision space. It is also attractive that with using only the direction of each value in x can achieve a high success attack rate!

2.3.2. PGD algorithm

PGD algorithm is formulated as the following, which is iteratively adjust the adversarial example by a small step α and clamp the value that out of the range of ε.

$$x'_{t+1} = Proj_\varepsilon\left(x'_t + \alpha \cdot sign\left(\nabla_x \mathcal{L}_\theta(x'_t, y)\right)\right)$$

$$x'_{t+1} = Proj_\varepsilon(x'_t - \alpha \cdot sign\left(\nabla_x \mathcal{L}_\theta(x'_t, y')\right))$$

These two algorithms can achieve untargeted and targeted attacks, respectively. 2.3.3. MI-FGSM algorithm

2.3.3. MI-FGSM algorithm

MI-FGSM algorithm is inspired by the introduction of momentum into SGD optimization algorithm. It can be considered as updating x with a SGDM algorithm with batch-size of 1. With the momentum, the calculated gradient will be more stable can will not drop into a bad local minimum.

$$x'_{t+1} = Clip_\varepsilon\left(x'_t + \alpha \cdot sign(g_{t+1})\right)$$

$$g_{t+1} = \xi \cdot g_t + \frac{\nabla_x \mathcal{L}_\theta(x'_t, y)}{||\nabla_x \mathcal{L}_\theta(x'_t, y)||_1}$$

2.3.4. DeepFool algorithm

DeepFool can find the closest distance of the given data to the decision boundary and add a minimum perturbation in L_2 considering a linear affine classifier. The perturbation δ can be iteratively calculated by:

$$\underset{\delta_t}{\mathrm{argmin}} ||\delta_t||_2 \; s.t. f(x'_t) + \nabla (x'_t)^T \delta_t$$

The iteration will stop once the $f(x'_t) \neq f(x)$, which means the attack is successful. This attack can also be generalized to multi-class scenario with general L_p, $p \in [0, \infty)$. Compare to the JSMA algorithm which reduce the elements for permutation number, DeepFool can reduced permutation intensity.

The setting of the attack algorithm for this study is summed up in Table 1. For the targeted attack, the target label will all be set to the class of tench.

Algorithm	**Mode**	**Constraint**	**Hyperparameters**
FGSM	untargeted	L_∞	$\varepsilon = 0.1$
PGD	untargeted	L_∞	$\varepsilon = 0.02, \alpha = 0.002, \text{iteration} = 20$
MI-FGSM	untargeted	L_∞	$\varepsilon = 0.02, \alpha = 0.002, \text{iteration} = 20, \text{momentum} = 0.9$
DeepFool	untargeted	L_2	step=50, overshoot=0.02
PGD	targeted	L_∞	$\varepsilon = 0.03, \alpha = 0.002, \text{iteration} = 40$

Table 1. The attack setting

2.4. Experiments

First, the generated adversarial examples via different attacking algorithms will directly be used to attack the original model which is utilized to generated them. Then, they will used to attack other models to test the transferability.

To investigate the embedding from middle layer of model, the embedding of both original image and the adversarial examples will be both extracted from different layers and projected to 2D space via t-SNE algorithm for visualization. The layer of ResNet series to be extracted is illustrated in Figure 1S. For DenseNet-121, only the layer before the last fully-connected layer will be extracted.

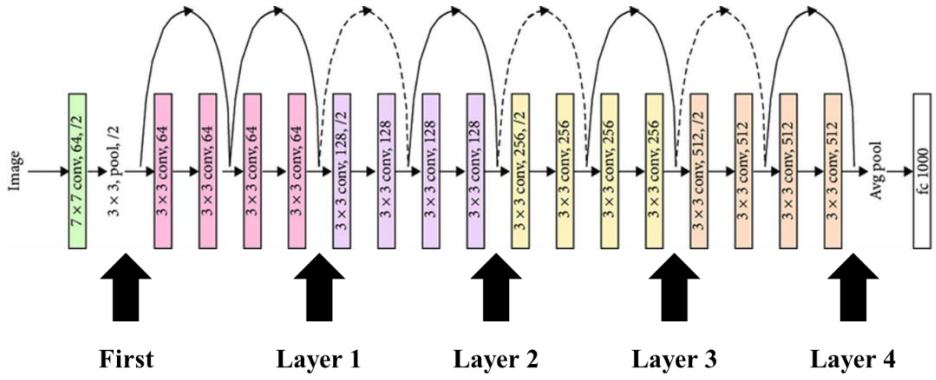


Figure 1S. The position where embedding is going to be extracted.

Finally, by comparing the distribution of these adversarial examples and original images, the most vulnerable layer can be found out easily.

3. Result

3.1. Dataset

Table 2 shows the number of training and validation data of each class of ImageNette and Figure 1 shows some examples of each class. All the images are re-sized to 224*224 pixel.

	Training	**Validation**
Tench	963	387
English springer	955	395
Cassette player	993	357
Chain saw	858	386
Church	941	409
French horn	956	294
Garbage truck	961	389
Gas pump	931	419
Golf ball	951	399
Parachute	960	390
Total	9469	3925

Table 2. The number of images of training and validation data.

Tench

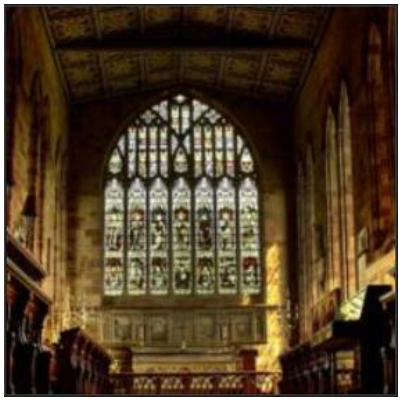
English

springer

Cassette player

Chain saw

Church



French horn

Garbage truck

Gas pump

Golf ball

Parachute

Figure 1. Some examples of each class in ImageNette.

3.2. Model training

ResNet-18, ResNet-50, DenseNet-121, and wide ResNet-50 are trained with the above training data and validated by those validation data. The accuracies are shown in Table 3. As expectation, DenseNet-121 achieves the highest accuracy in validation set following by ResNet-50 and ResNet-18. Wide ResNet-50 v2 obtains the lowest accuracy of 97.40%.

	Training set (%)	**Validation set (%)**
ResNet-18	100	97.66
ResNet-50	99.93	97.83
DenseNet-121	99.98	98.14
Wide ResNet-50 v2	99.83	97.40

Table 3. The accuracy achieved by each model.

3.3. Adversarial examples generation

The adversarial examples are generated with four models via 4 attacking algorithm and the attacking results are shown in Table 4, which represents the accuracy against the adversarial examples. Regarding the targeted attack, among all the attacking algorithms, FGSM, which only generate the adversarial examples with one step, cannot attack the model very well. While the DeepFool shows a better performance, PGD and MI-FGSM demonstrates the best attacking power and even lessen the accuracy under 10% against the ResNet-18 and DenseNet-121 models. For the targeted attack, it is hard to make all the adversarial examples be predicted as the class of tench. However, with the powerful PGD algorithm, there are still 70 to 80% images are successfully classified as tench and achieves the targeted attack.

Some adversarial examples with the difference comparing with the original images are provided in Figure 2. All the perturbed noise in adversarial examples provided are imperceptible to human; that's, it's impossible for human to distinguish which image is the adversarial one. By observing the difference of images generated by different attacking algorithm, FGSM shows a random perturbation while PGD and MI-FGSM show some mysterious patterns. DeepFool, which perturbs images with L_2 constraint illustrates many continuous patterns which are distinct from other methods with L_∞ constraint.

	Untargeted				Targeted
	FGSM	PGD	MI-FGSM	DeepFool	PGD
ResNet-18	26.45	8.79	8.20	20.74	19.62
ResNet-50	33.10	14.93	14.42	24.28	26.42
DenseNet-121	33.22	8.03	7.54	21.91	18.29
Wide ResNet-50 v2	33.78	17.25	16.82	27.24	29.63

Table 4. Accuracy against adversarial examples.

FGSM untargeted attack

Original Adversarial Difference (x5)

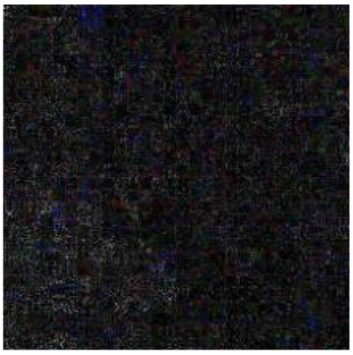
Tench



English

springer



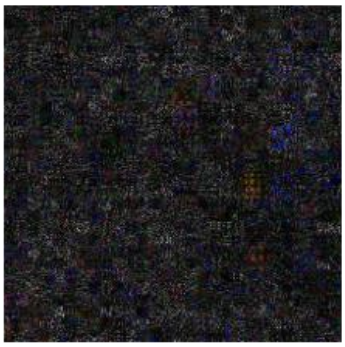


Cassette player



Chain saw



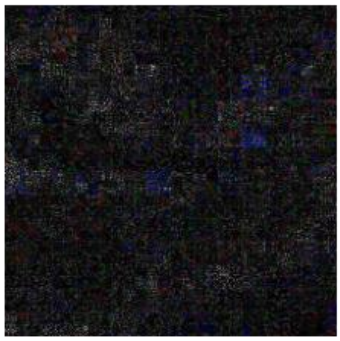


Church



French horn

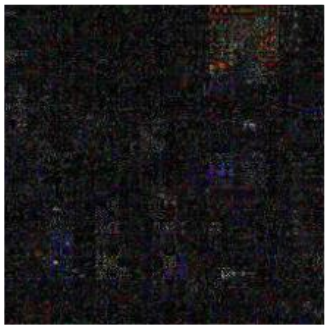




Garbage truck

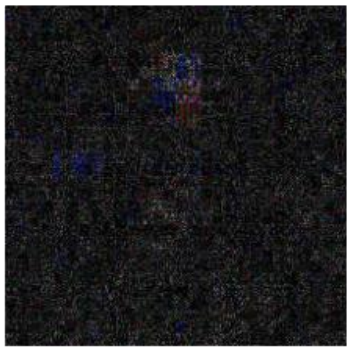


Gas pump



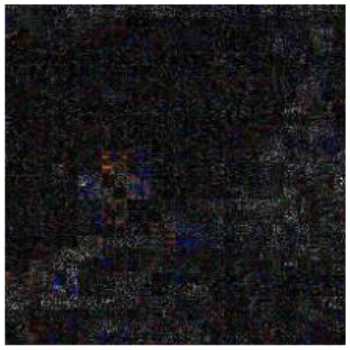
Golf ball



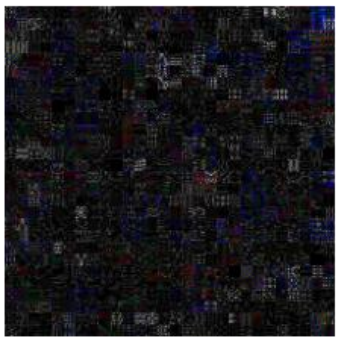
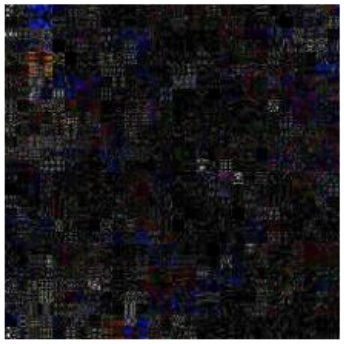
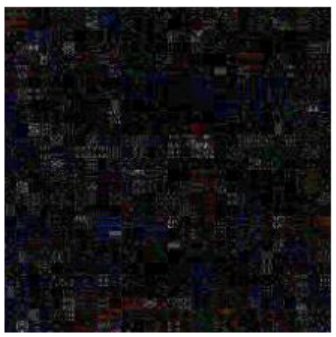
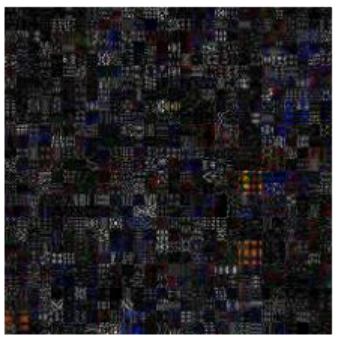


Parachute



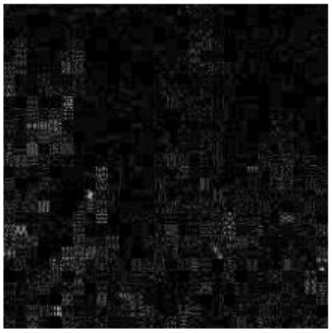


PGD untargeted attack

	Original	Adversarial	Difference (x10)
Tench			
English springer			
Cassette player			
Chain saw			

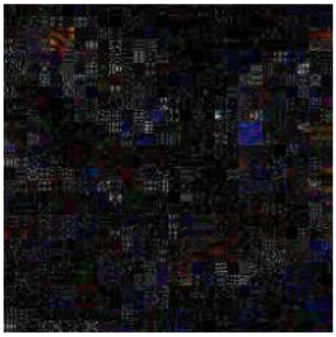
Church





French horn

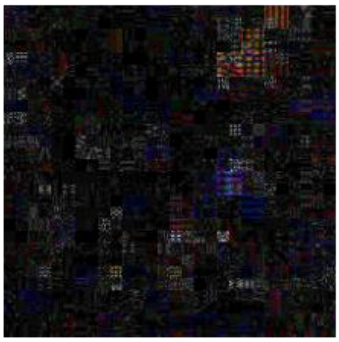




Garbage truck

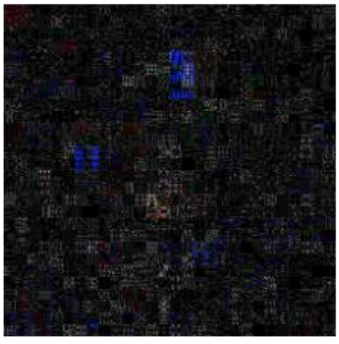


Gas pump



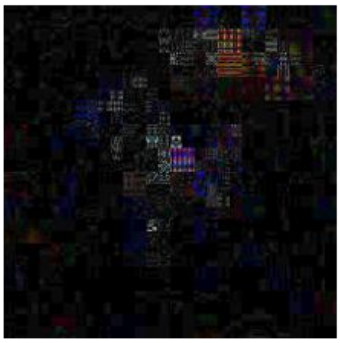
Golf ball



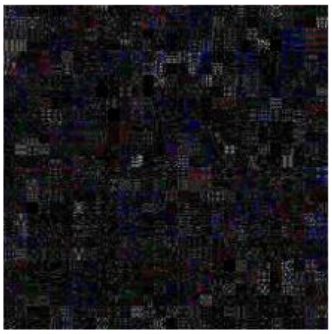
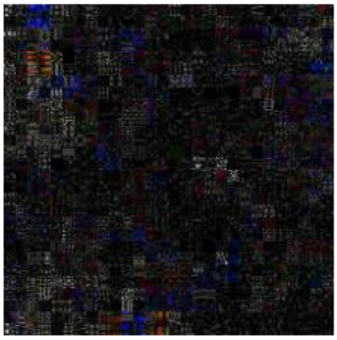
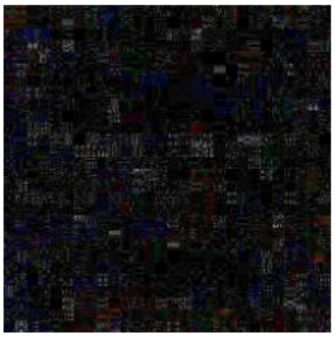


Parachute



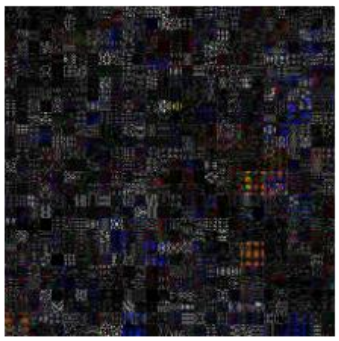


MI-FGSM untargeted attack

	Original	Adversarial	Difference (x10)
Tench			
English springer			
Cassette player			

Chain saw

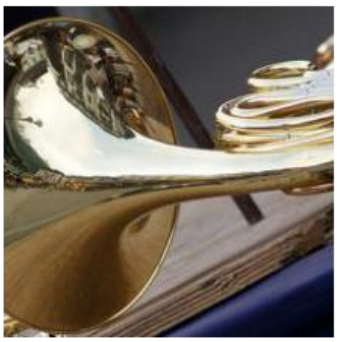
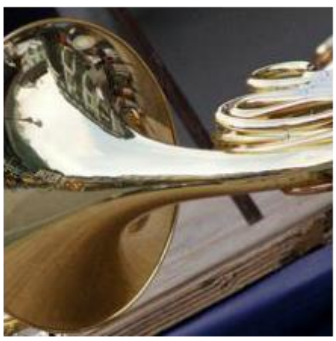
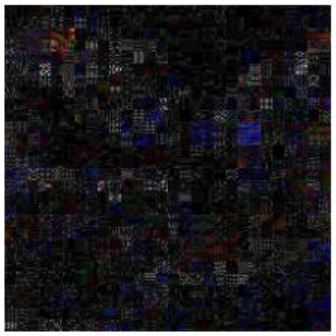




Church

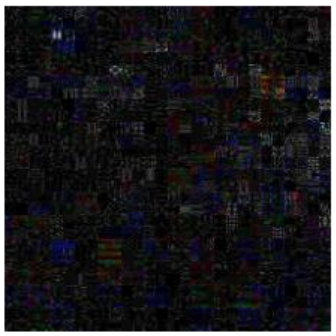


French horn

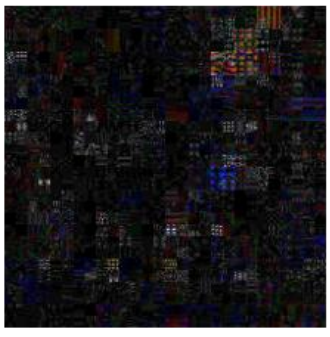




Garbage truck



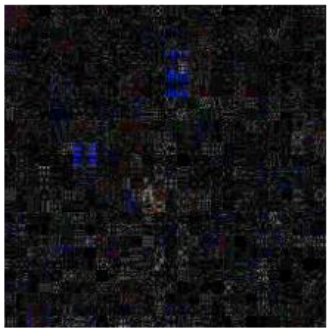


Gas pump



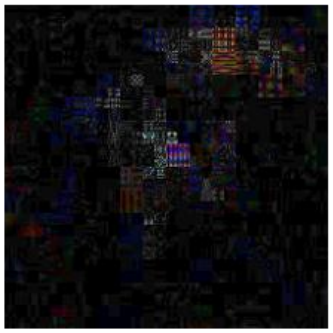
Golf ball



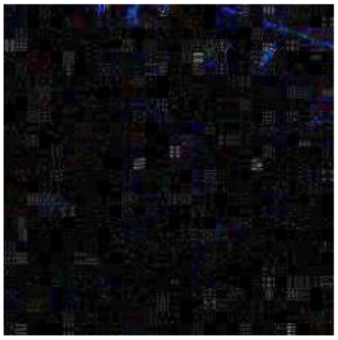
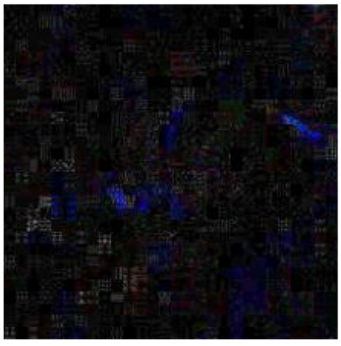


Parachute





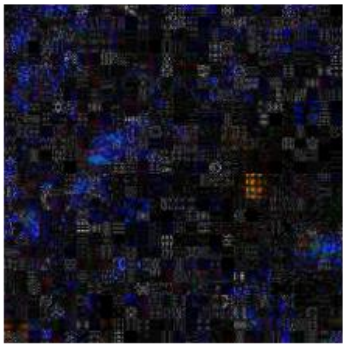
DeepFool untargeted attack

	Original	Adversarial	Difference (x8)
Tench			
English springer			

Cassette player

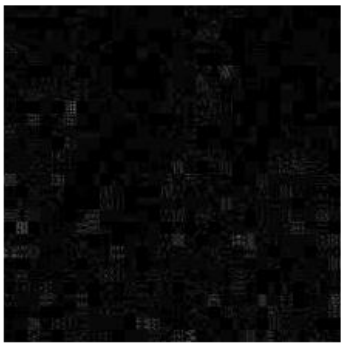
Chain saw



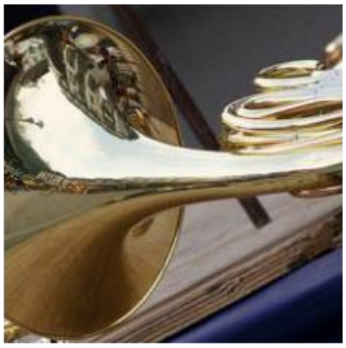
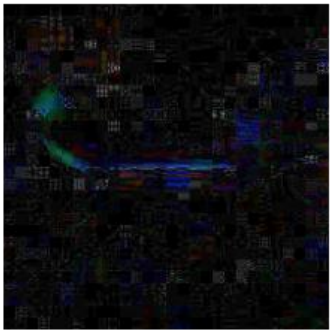


Church





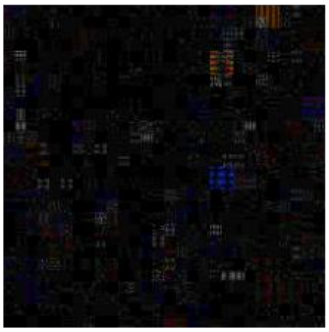
French horn

Garbage truck

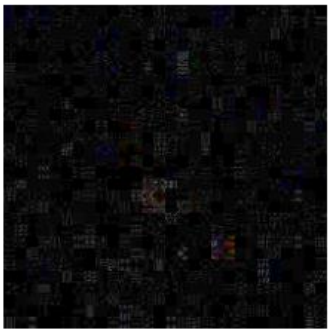


Gas pump



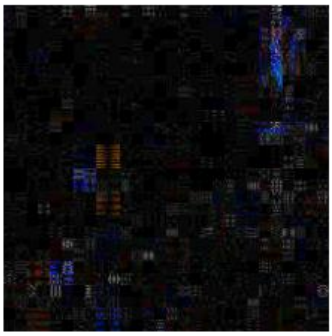
Golf ball



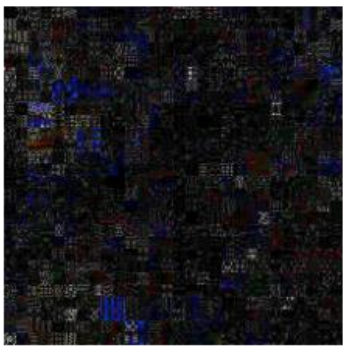


Parachute





PGD targeted attack

	Original	Adversarial	Difference (x10)
English springer			

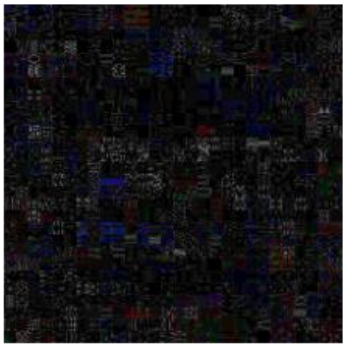
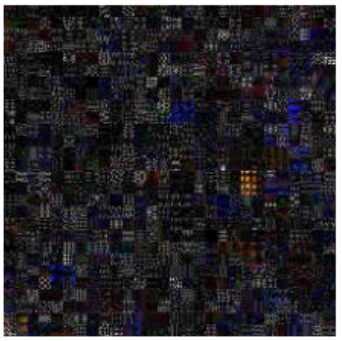
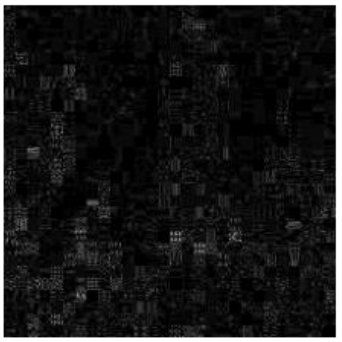
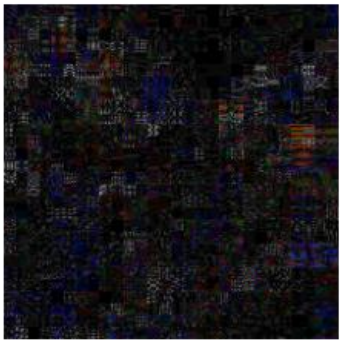
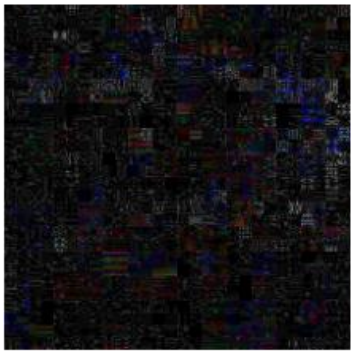
Cassette player			
Chain saw			
Church			
French horn			
Garbage truck			

Figure 2. Some examples of adversarial examples, which are all imperceptible to human.

3.4. Transferability

The transferability of adversarial examples generated by FGSM, PGD, MI-FGSM, and DeepFool algorithms against ResNet-18, ResNet-50, DenseNet-121, and wide ResNet-50 is shown in Table 4, 5, 6, and 7, respectively. It can be noticed that even if the attacking capability of FGSM is not as powerful as other iterative-based methods, it has the most transferability and might be owing to the specificity of adversarial examples generated by iterative-based methods. However, generally all the transferability is not good enough and make the attack futile, which might be caused by the small perturbation limit that initially set.

From\to	ResNet-18	ResNet-50	DenseNet-121	wide ResNet-50
ResNet-18	-	90.95	90.98	91.88

ResNet-50	90.36	-	90.63	90.75
DenseNet-121	89.52	90.42	-	90.86
wide ResNet-50	91.31	89.95	91.43	-

Table 4. Transferability of adversarial examples generated by FGSM.

From\to	ResNet-18	ResNet-50	DenseNet-121	wide ResNet-50
ResNet-18	-	97.45	98.17	96.90
ResNet-50	97.39	-	98.00	96.65
DenseNet-121	97.54	97.40	-	96.86
wide ResNet-50	97.68	97.14	98.22	-

Table 5. Transferability of adversarial examples generated by PGD.

From\to	ResNet-18	ResNet-50	DenseNet-121	wide ResNet-50
ResNet-18	-	97.44	98.06	97.01
ResNet-50	97.34	-	97.95	96.67
DenseNet-121	97.41	97.38	-	96.77
wide ResNet-50	97.69	97.09	98.20	-

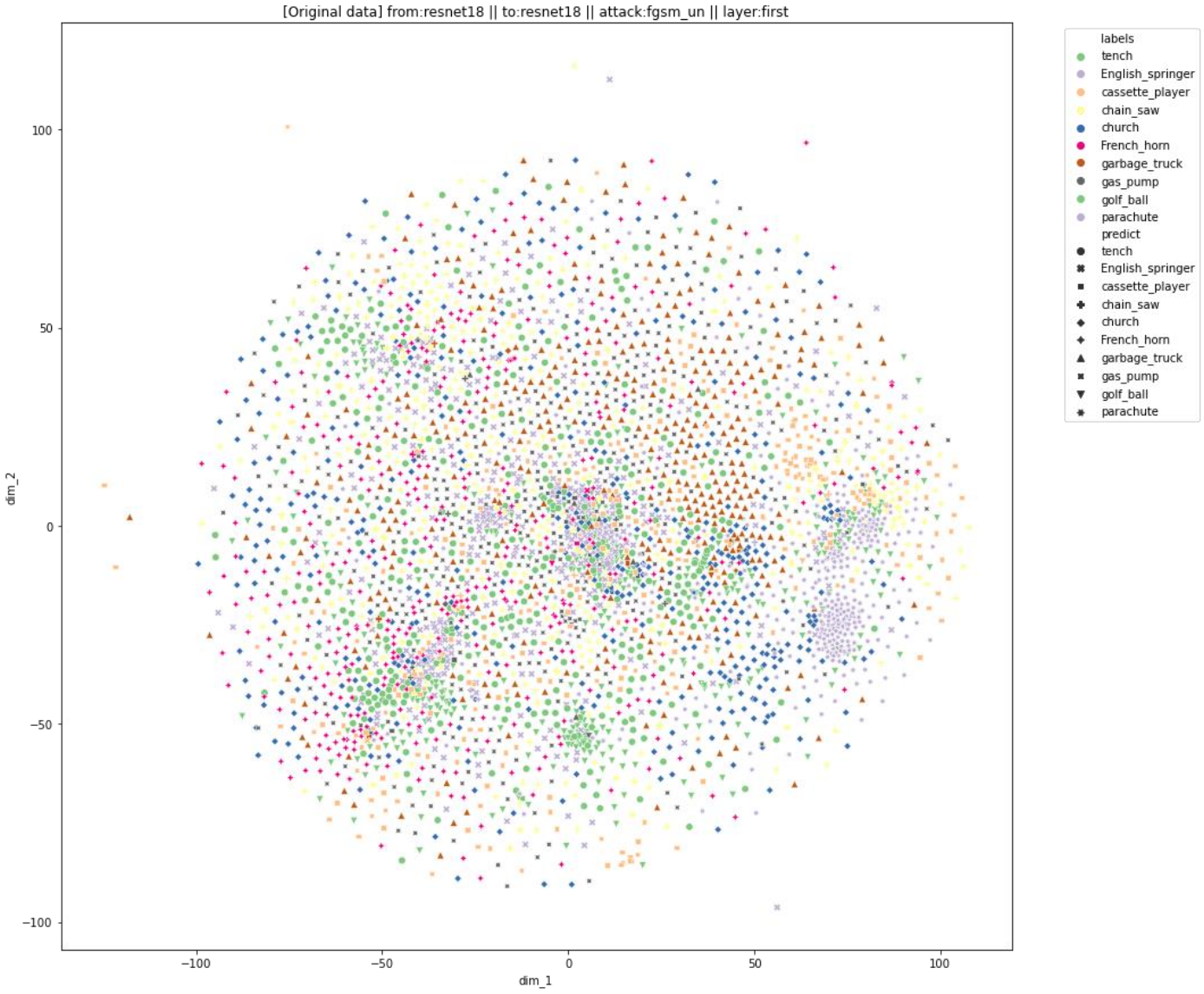
Table 6. Transferability of adversarial examples generated by MI-FGSM.

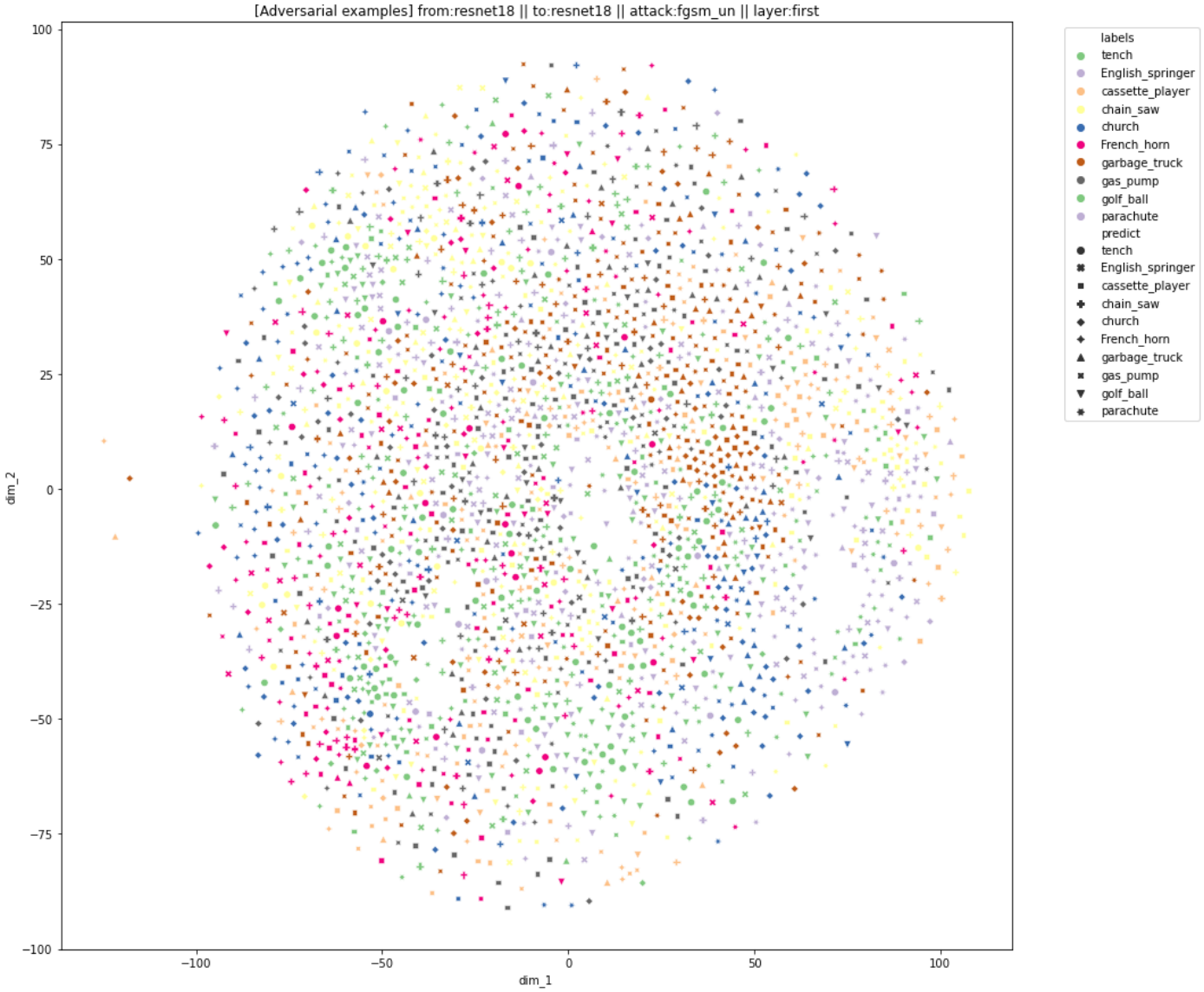
From\to	ResNet-18	ResNet-50	DenseNet-121	wide ResNet-50
ResNet-18	-	97.71	98.38	97.12
ResNet-50	97.64	-	98.16	96.85
DenseNet-121	97.73	97.69	-	96.96
wide ResNet-50	97.75	97.69	98.26	-

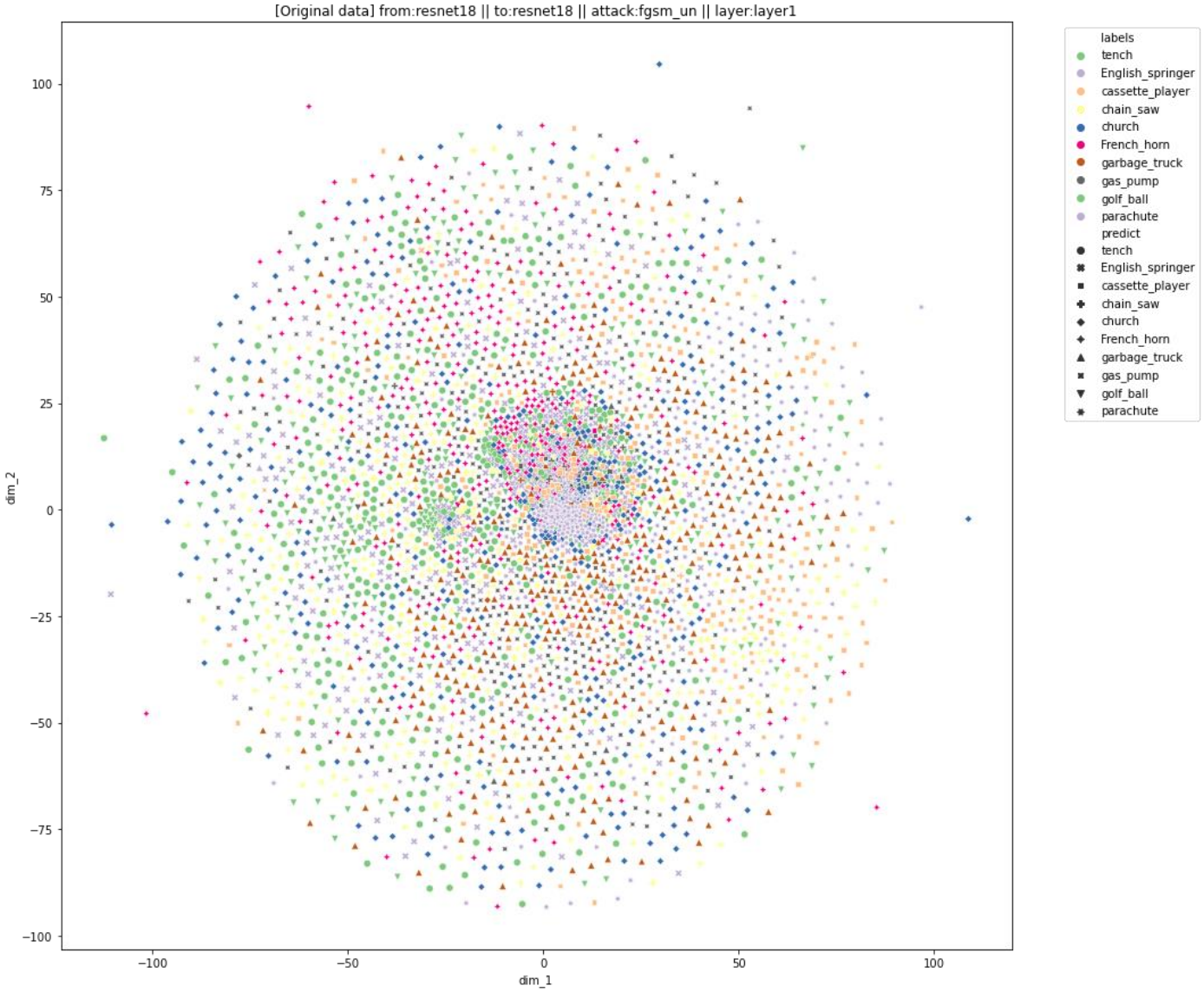
Table 7. Transferability of adversarial examples generated by DeepFool.

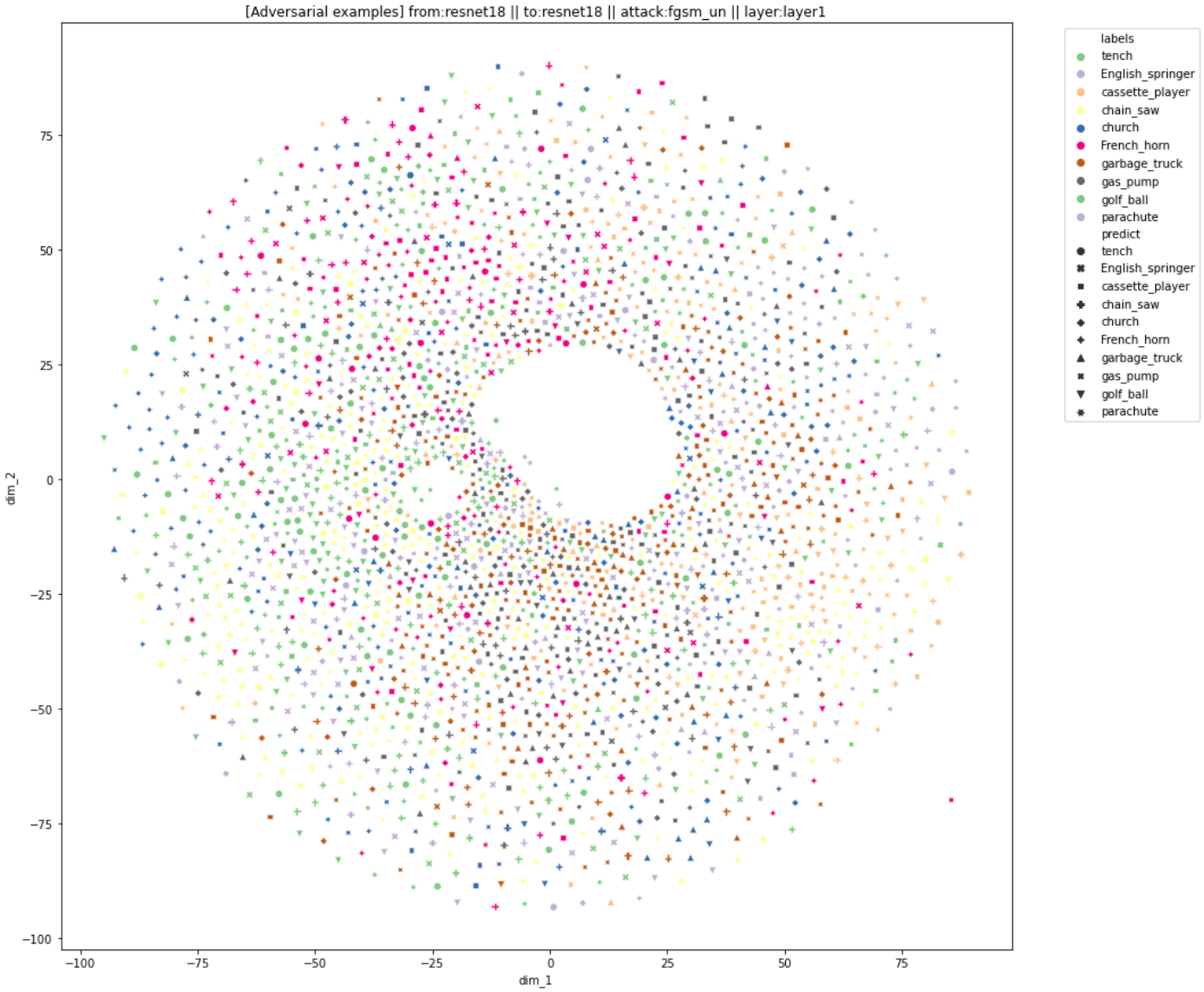
3.5. Embedding analysis of untargeted attack

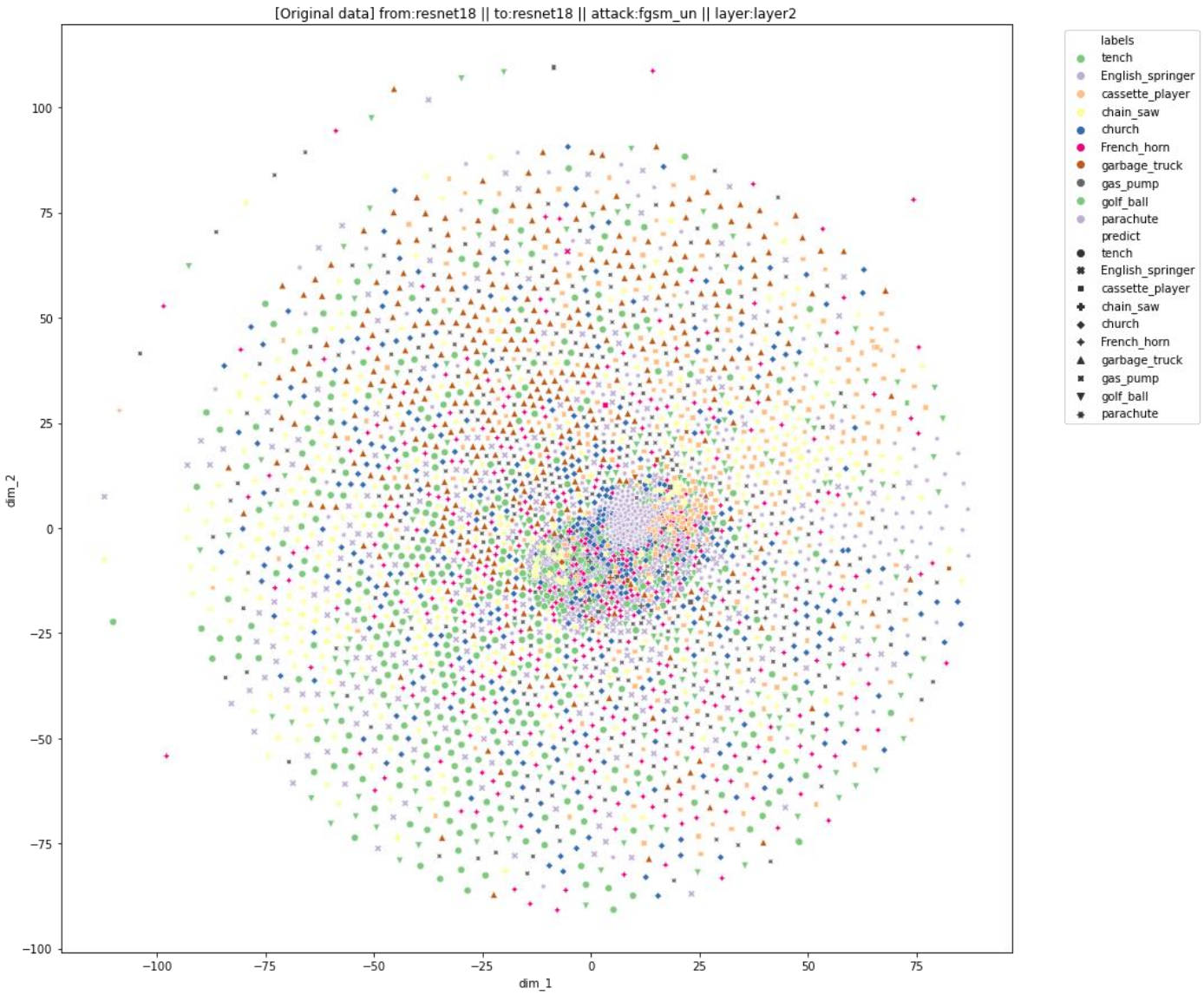
The scenario of adversarial examples generated by ResNet-18 with FGSM algorithm against ResNet-18 model is first analyzed and the embedding extracted from each layer is visualized in Figure 3. In the first, layer-1, layer-2, and layer-3 layers, most of the original data will cluster in the middle or somewhere but the adversarial examples will not. However, the distribution of adversarial examples is similar to the original data outside the clustering centers. An extreme change occurs in layer-4, the original data are abruptly well-separated while the adversarial data are somehow far from the clustering center of each class of original data, which eventually make them misclassified. Something interesting is that, some adversarial examples are exactly close to the clustering center but are still misclassified, which might due to the critical decision boundary in hyperplane or the problem in the last fully-connected layer.

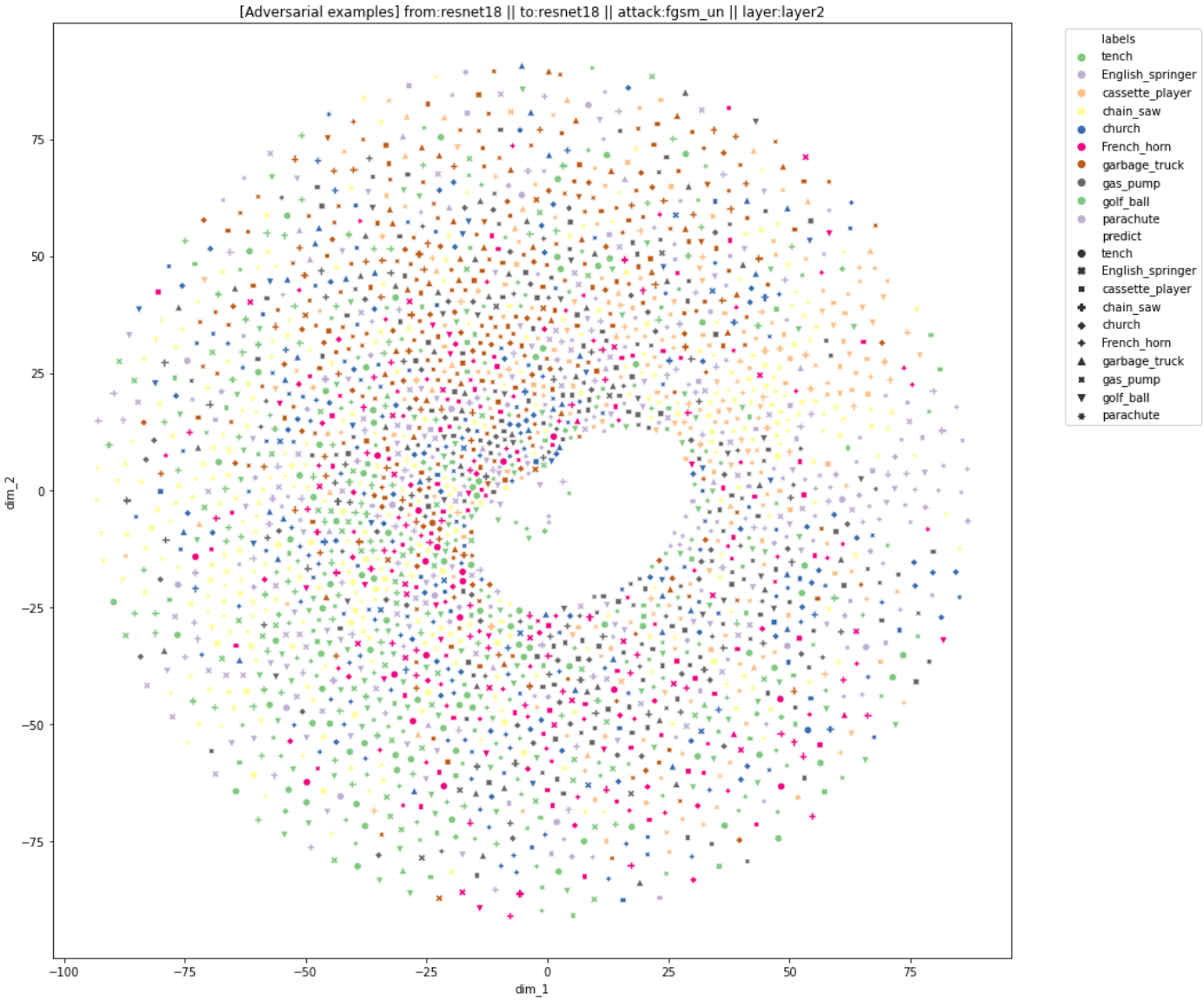


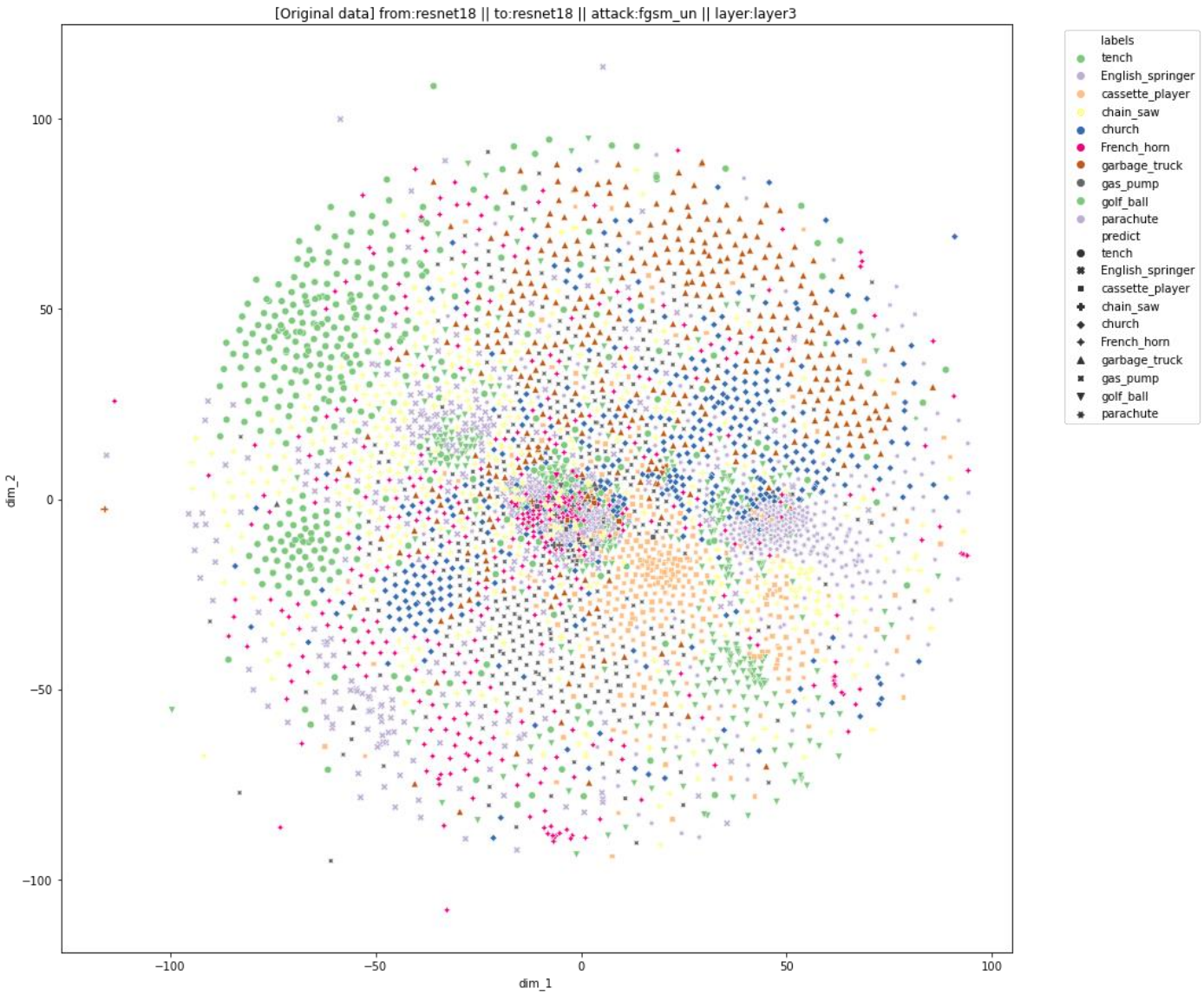

[Adversarial examples] from:resnet18 || to:resnet18 || attack:fgsm_un || layer:first
dim_1
dim_2
labels
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
predict
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute

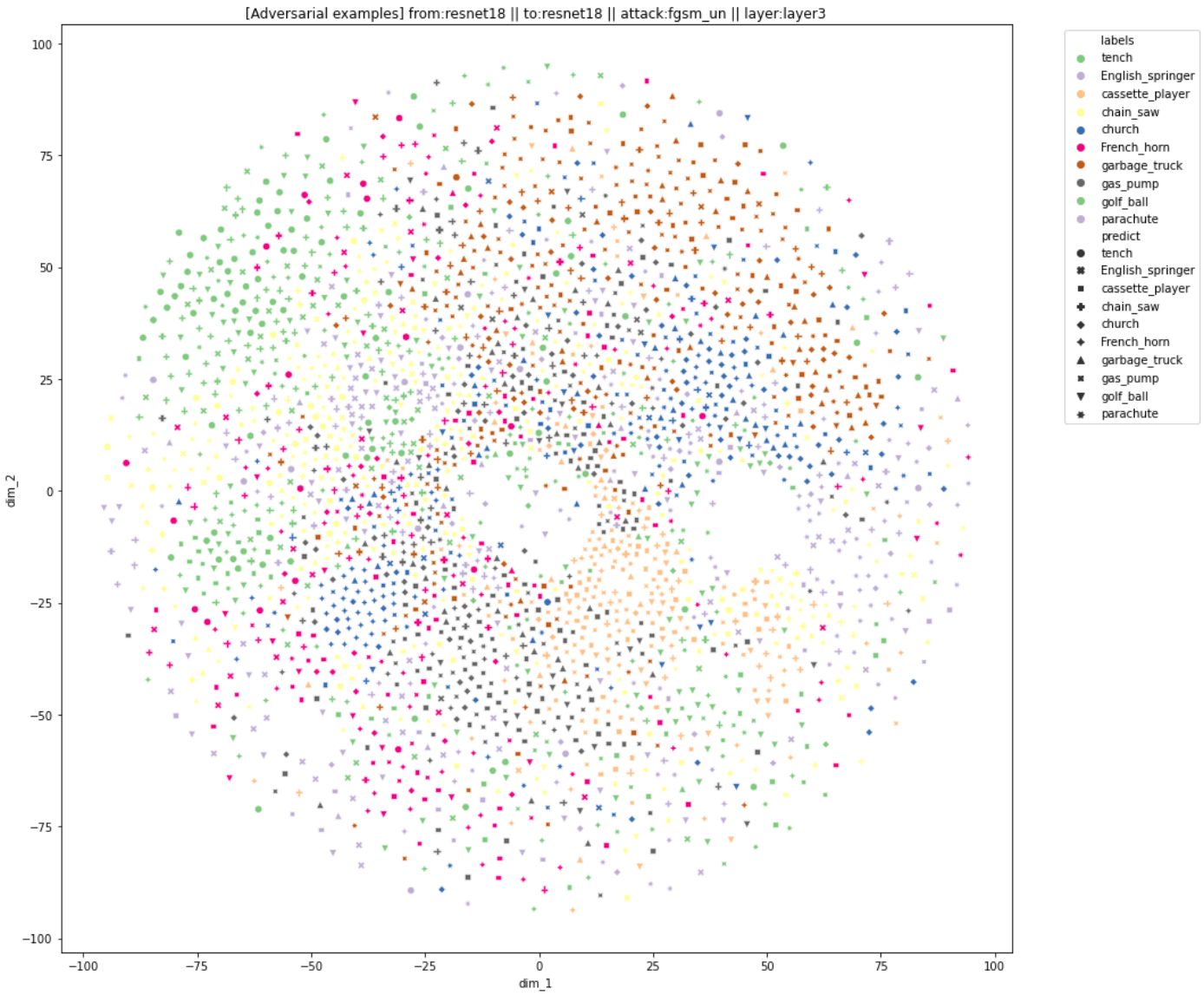

[Original data] from:resnet18 || to:resnet18 || attack:fgsm_un || layer:layer1
dim_1
dim_2
labels
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
predict
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute

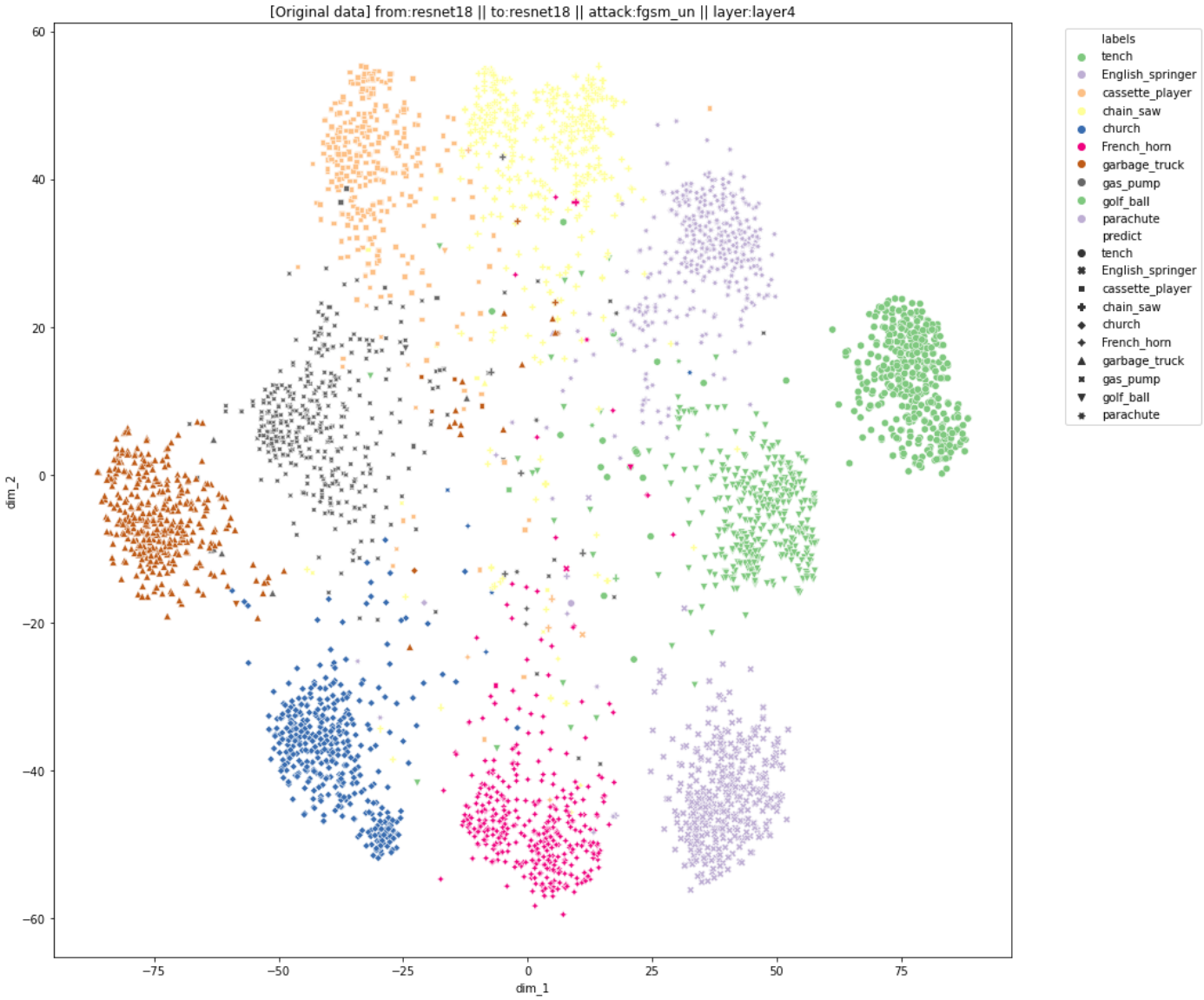

[Adversarial examples] from:resnet18 || to:resnet18 || attack:fgsm_un || layer:layer1
dim_1
dim_2
labels
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
predict
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute


[Original data] from:resnet18 || to:resnet18 || attack:fgsm_un || layer:layer2
dim_1
dim_2
labels
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
predict
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute


[Adversarial examples] from:resnet18 || to:resnet18 || attack:fgsm_un || layer:layer2
dim_1
dim_2
labels
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
predict
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute


[Original data] from:resnet18 || to:resnet18 || attack:fgsm_un || layer:layer3
dim_1
dim_2
100
50
0
−50
−100
−100
−50
0
50
100
labels
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
predict
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute


[Adversarial examples] from:resnet18 || to:resnet18 || attack:fgsm_un || layer:layer3
dim_1
dim_2
labels
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
predict
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute


[Original data] from:resnet18 || to:resnet18 || attack:fgsm_un || layer:layer4
dim_1
dim_2
labels
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
predict
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute

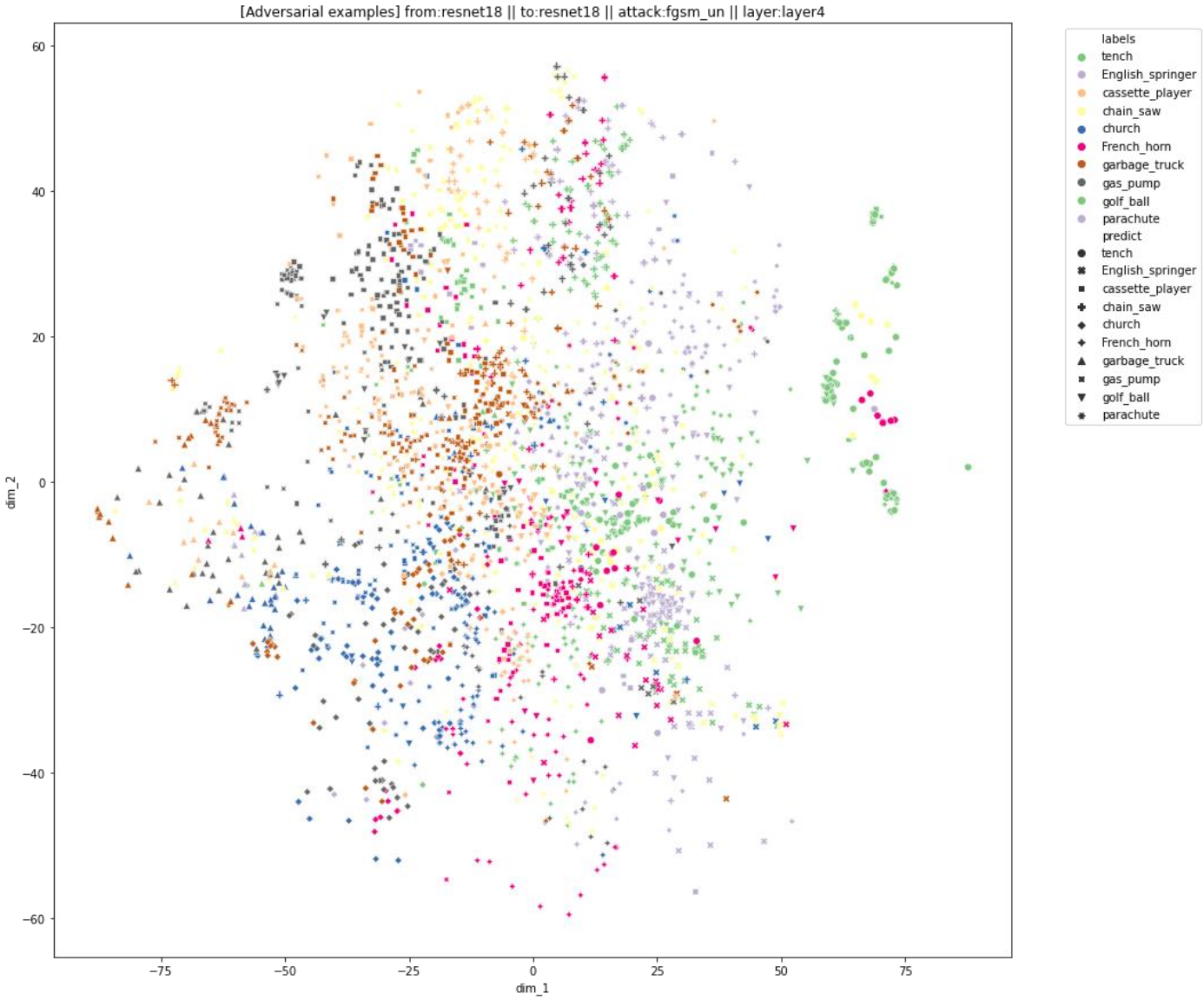
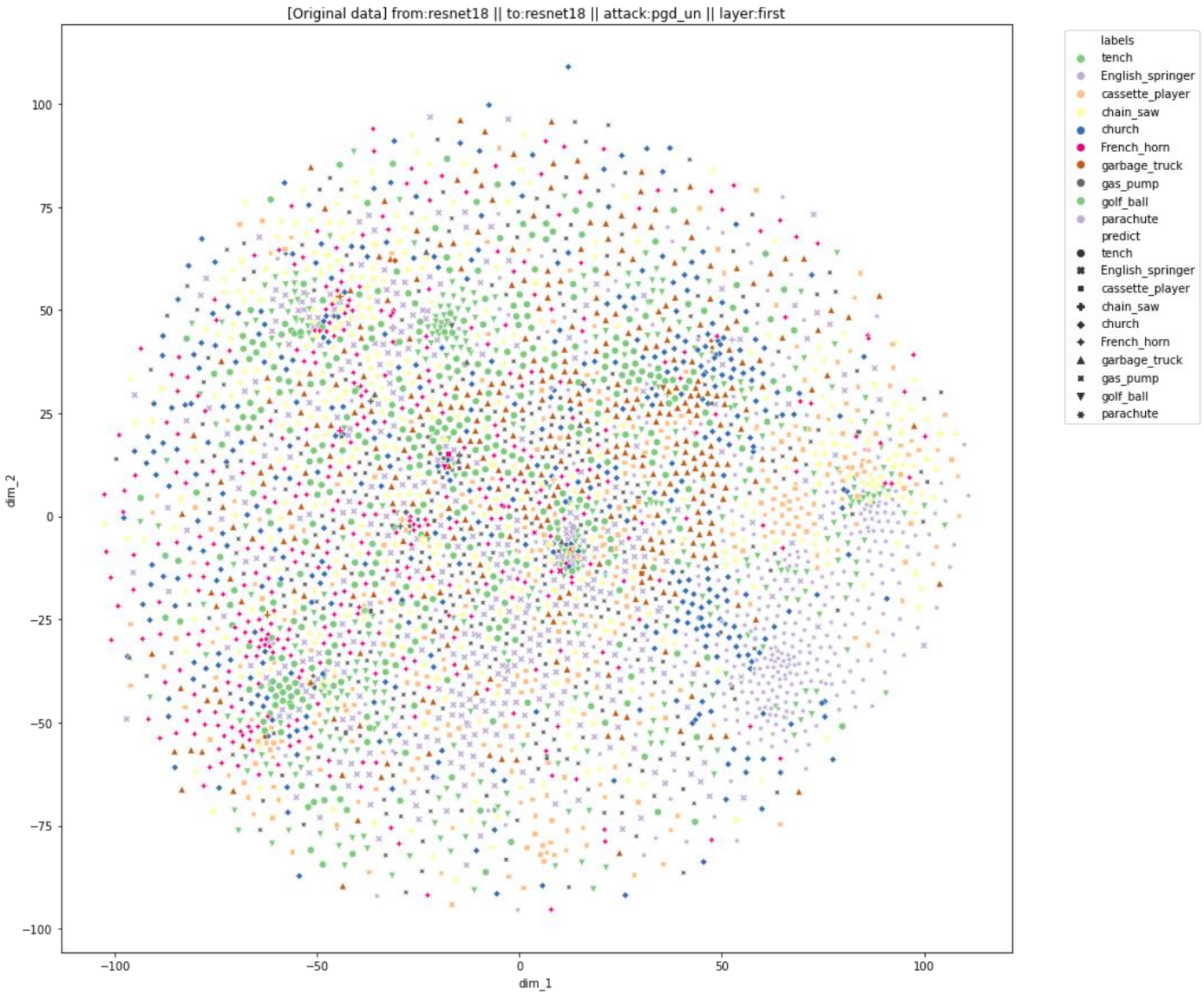
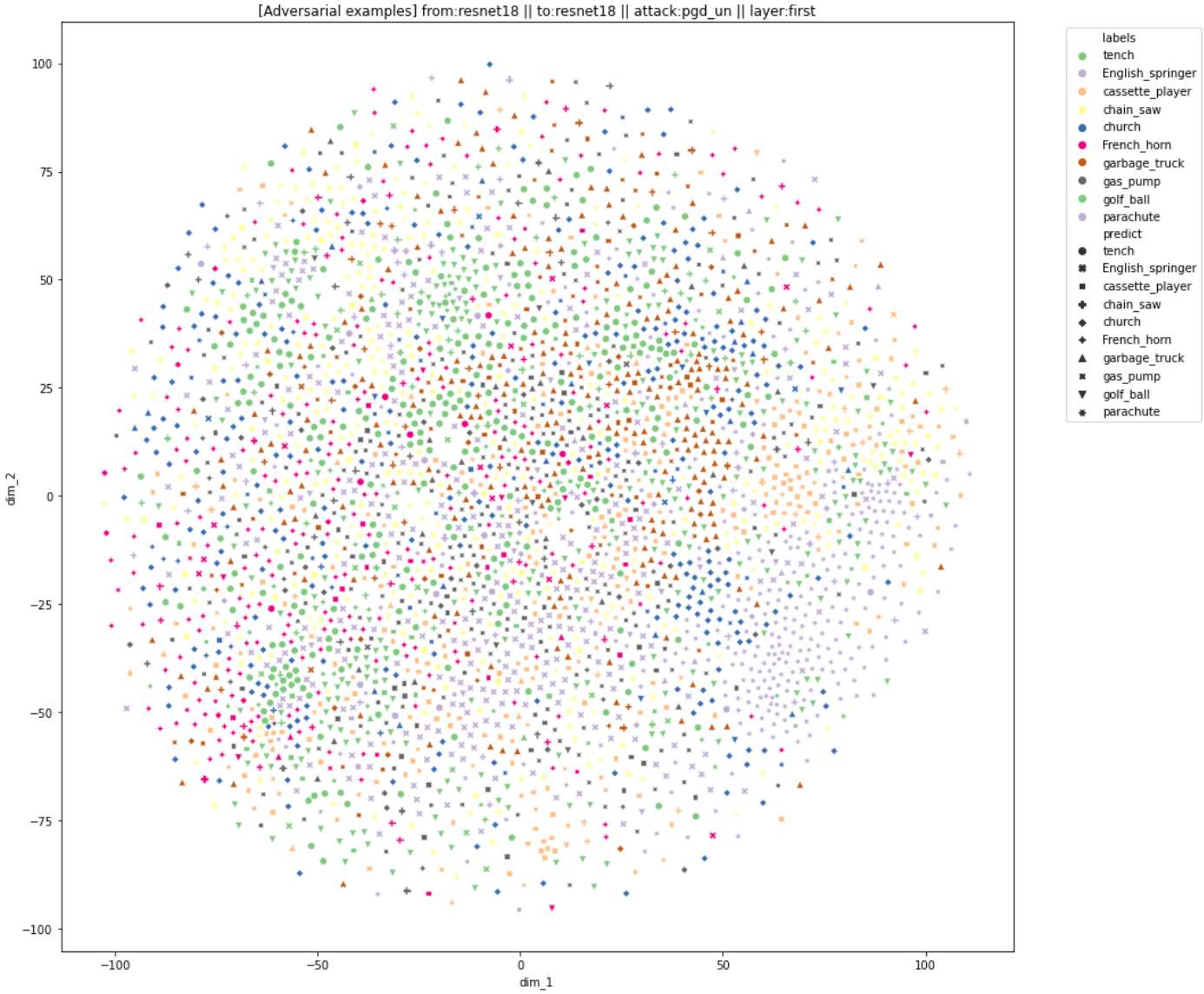
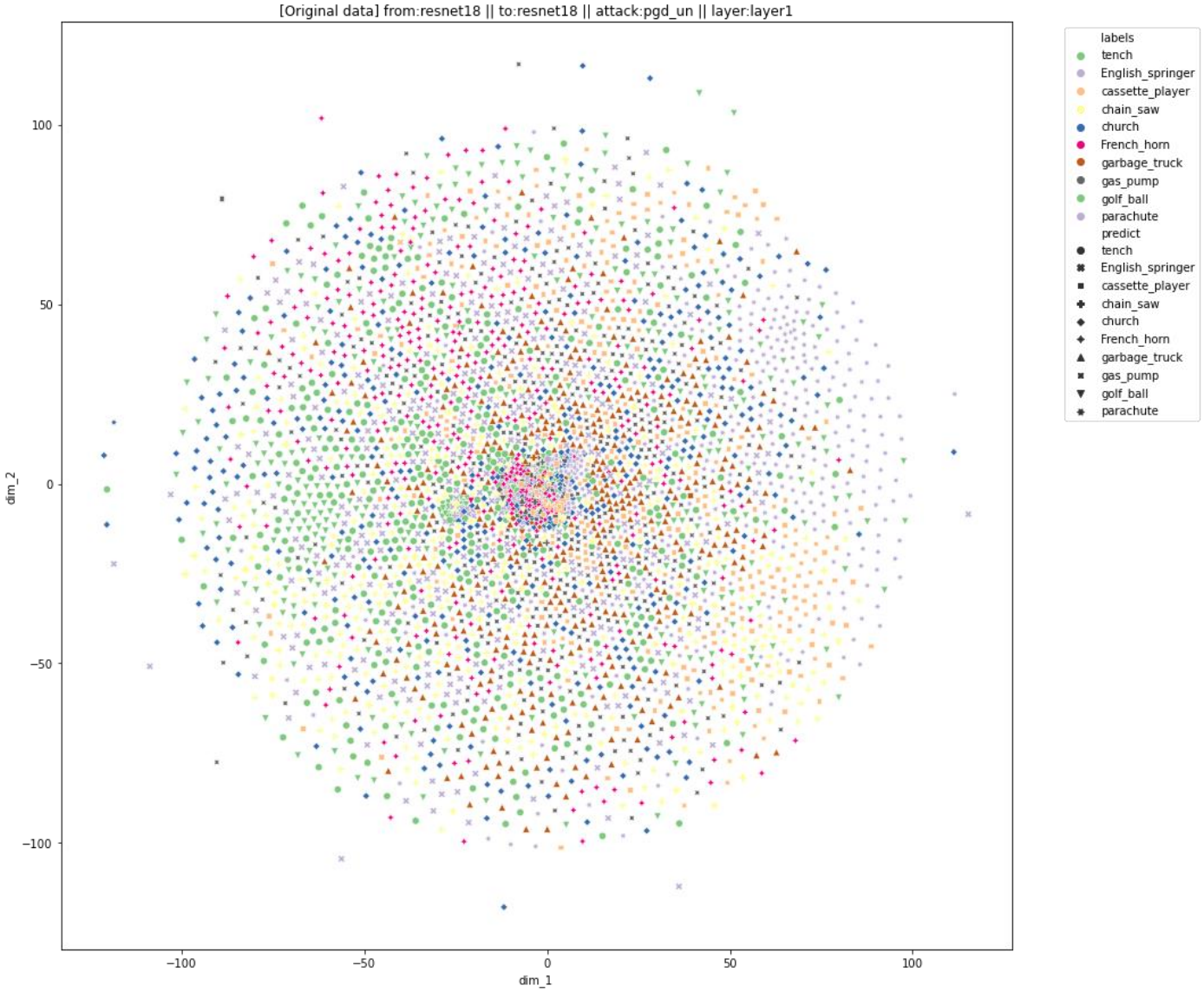


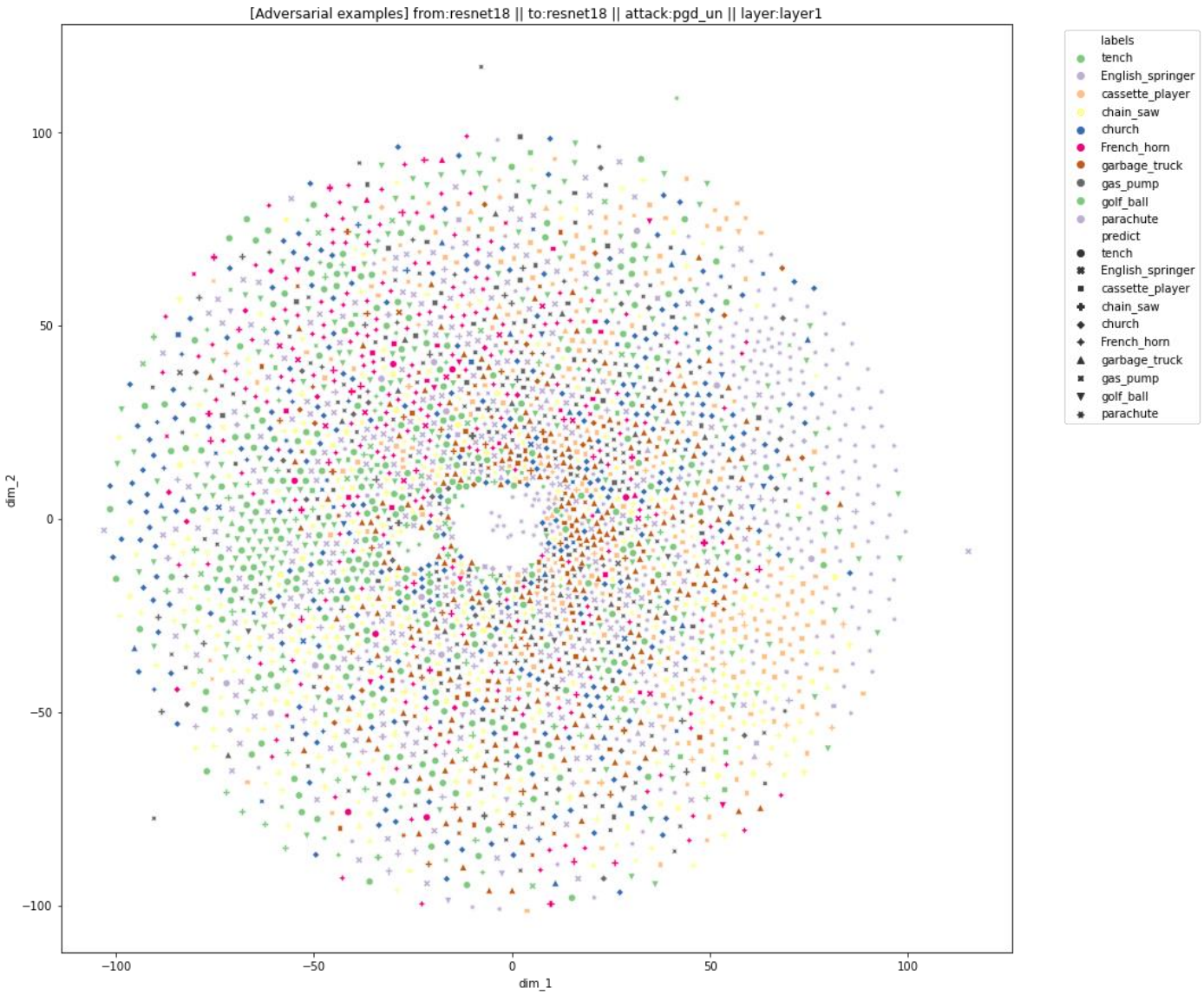
Figure 3. Embedding of adversarial examples generated by ResNet-18 with FGSM algorithm from each layer of ResNet-18.

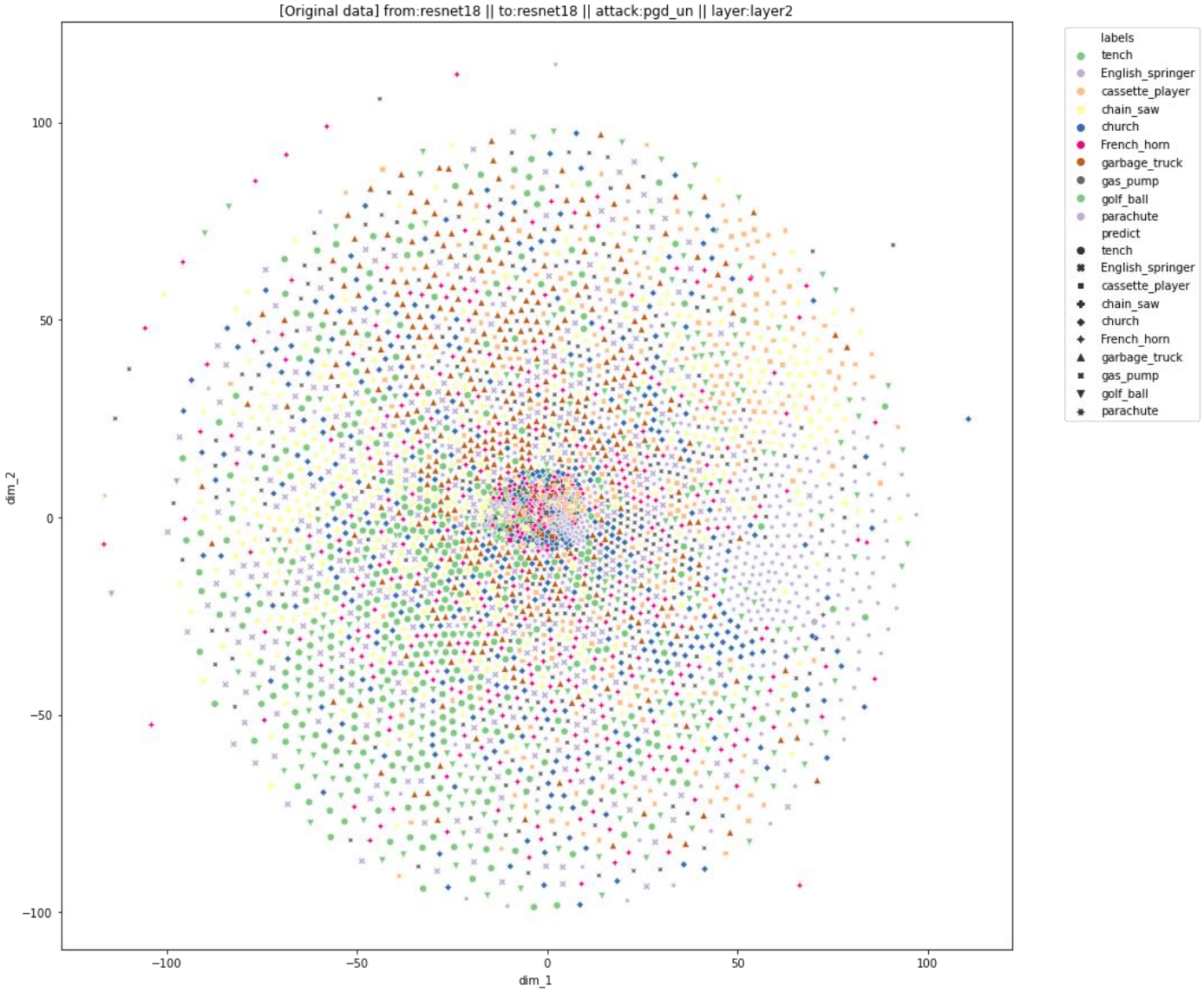
Another scenario of adversarial examples generated by ResNet-18 with PGD algorithm against ResNet-18 model is visualized in Figure 4. The behavior of first, layer-1, layer-2, and layer-3 layers are similar to the case of FGSM. Surprisingly, the embedding extracted from layer-4 show that the distribution of adversarial examples is congruous with the original naïve data, which might indicate that the PGD algorithm can really find the vulnerability in the decision boundary and what make the model vulnerable are layer-4 and the last fully-connected layer.

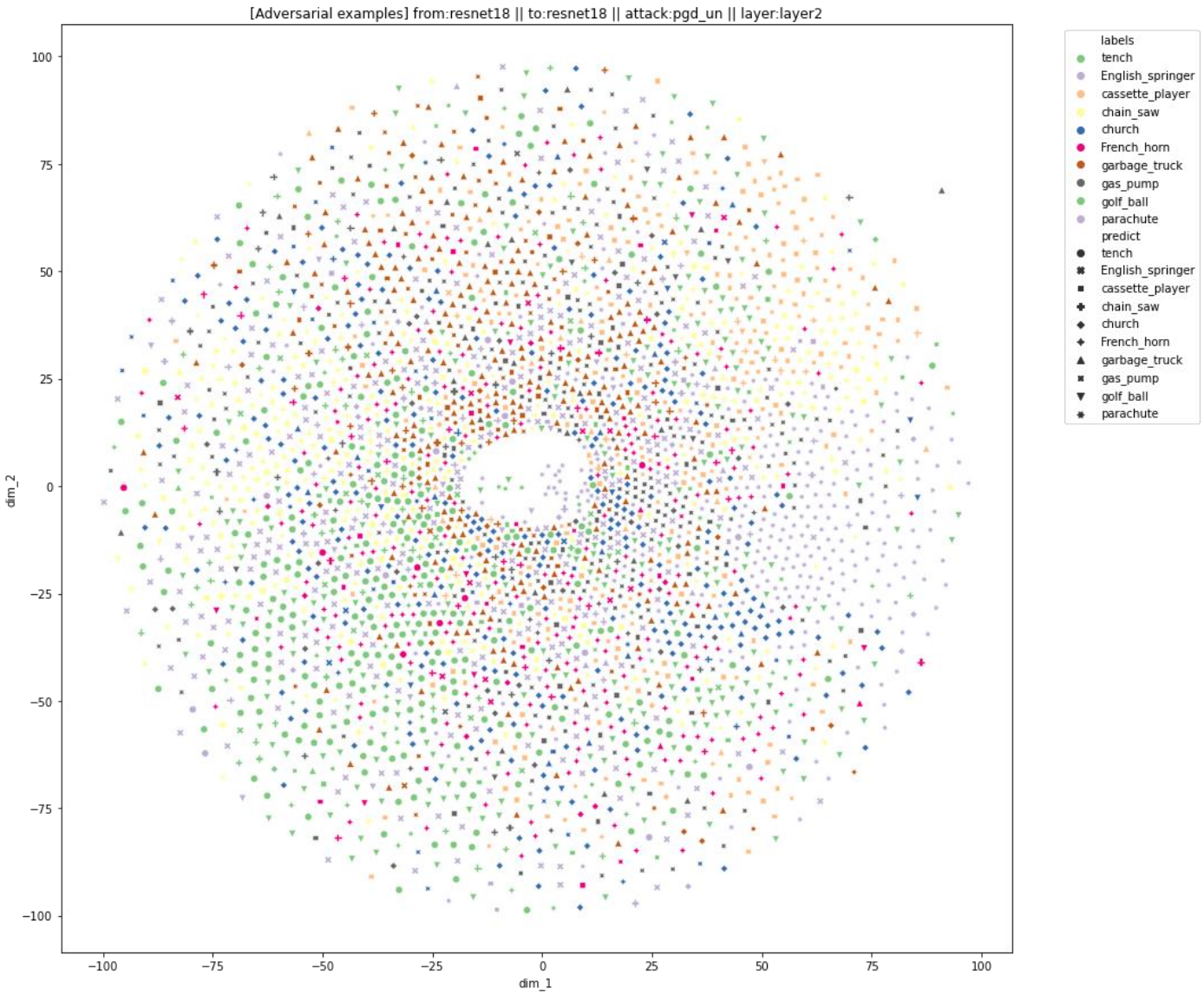

[Original data] from:resnet18 || to:resnet18 || attack:pgd_un || layer:first
dim_1
dim_2
labels
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
predict
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute

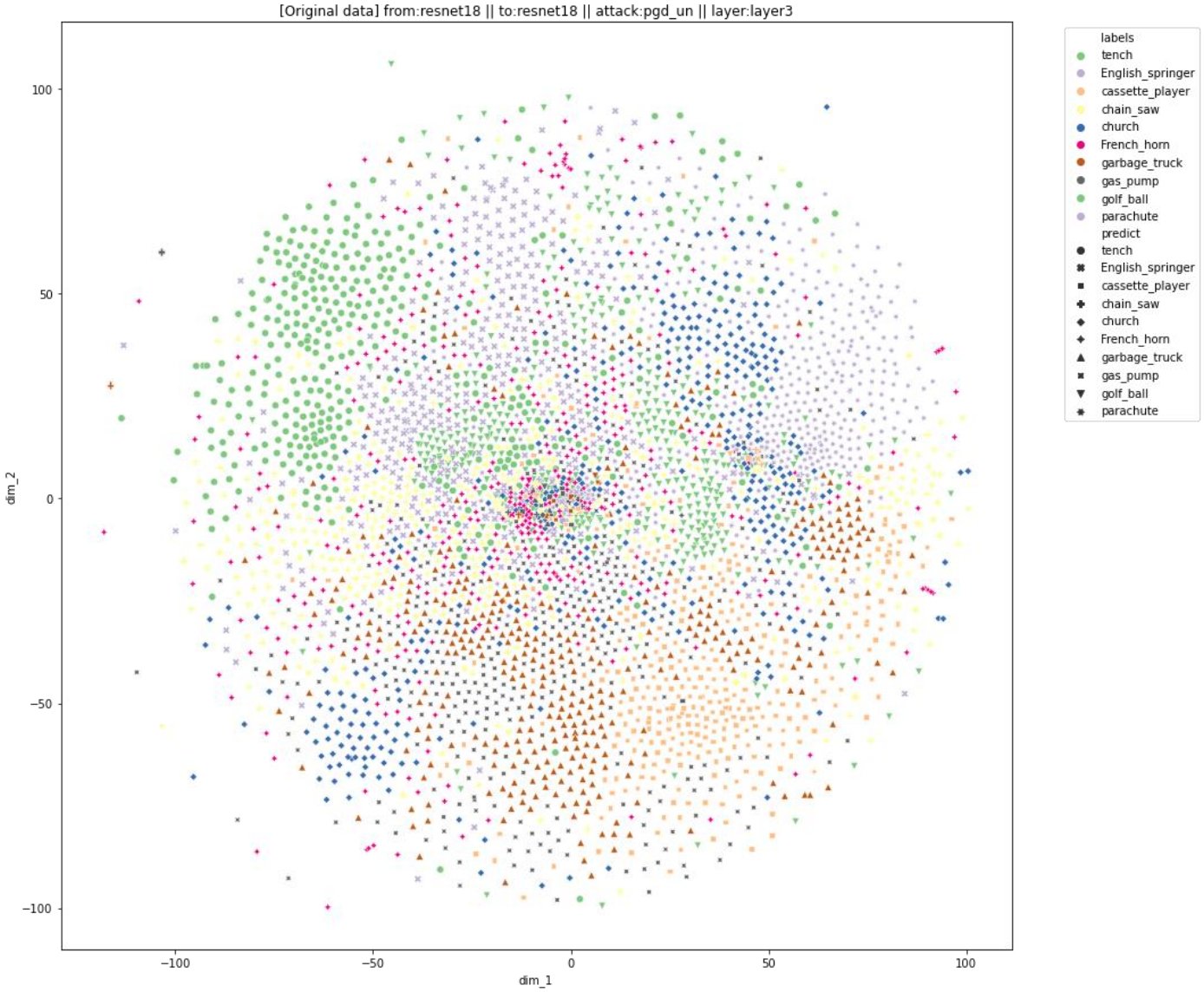

[Adversarial examples] from:resnet18 || to:resnet18 || attack:pgd_un || layer:first
dim_1
dim_2
labels
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
predict
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute

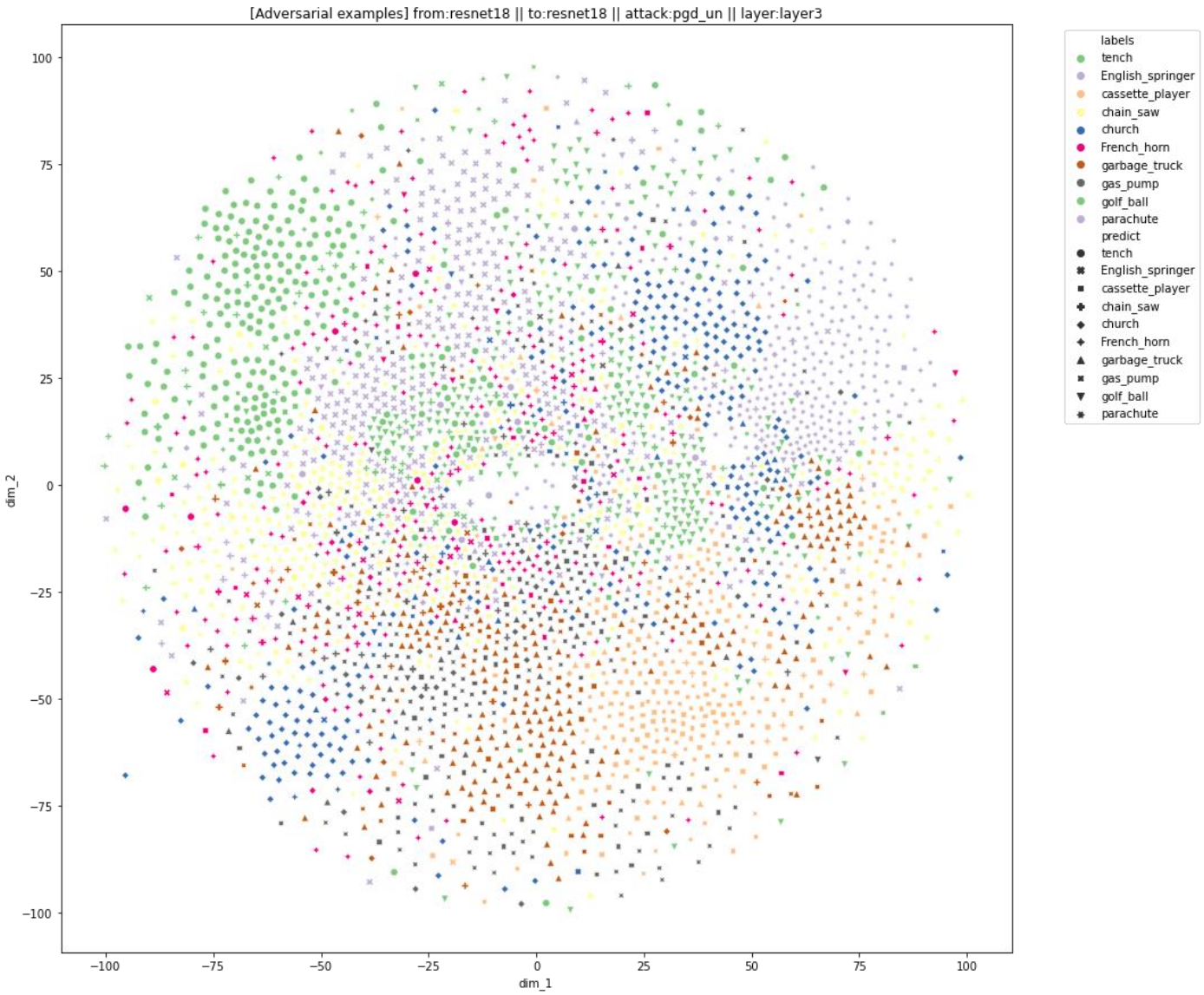

[Original data] from:resnet18 || to:resnet18 || attack:pgd_un || layer:layer1
labels
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
predict
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
dim_1
dim_2
100
50
0
−50
−100
−100
−50
0
50
100

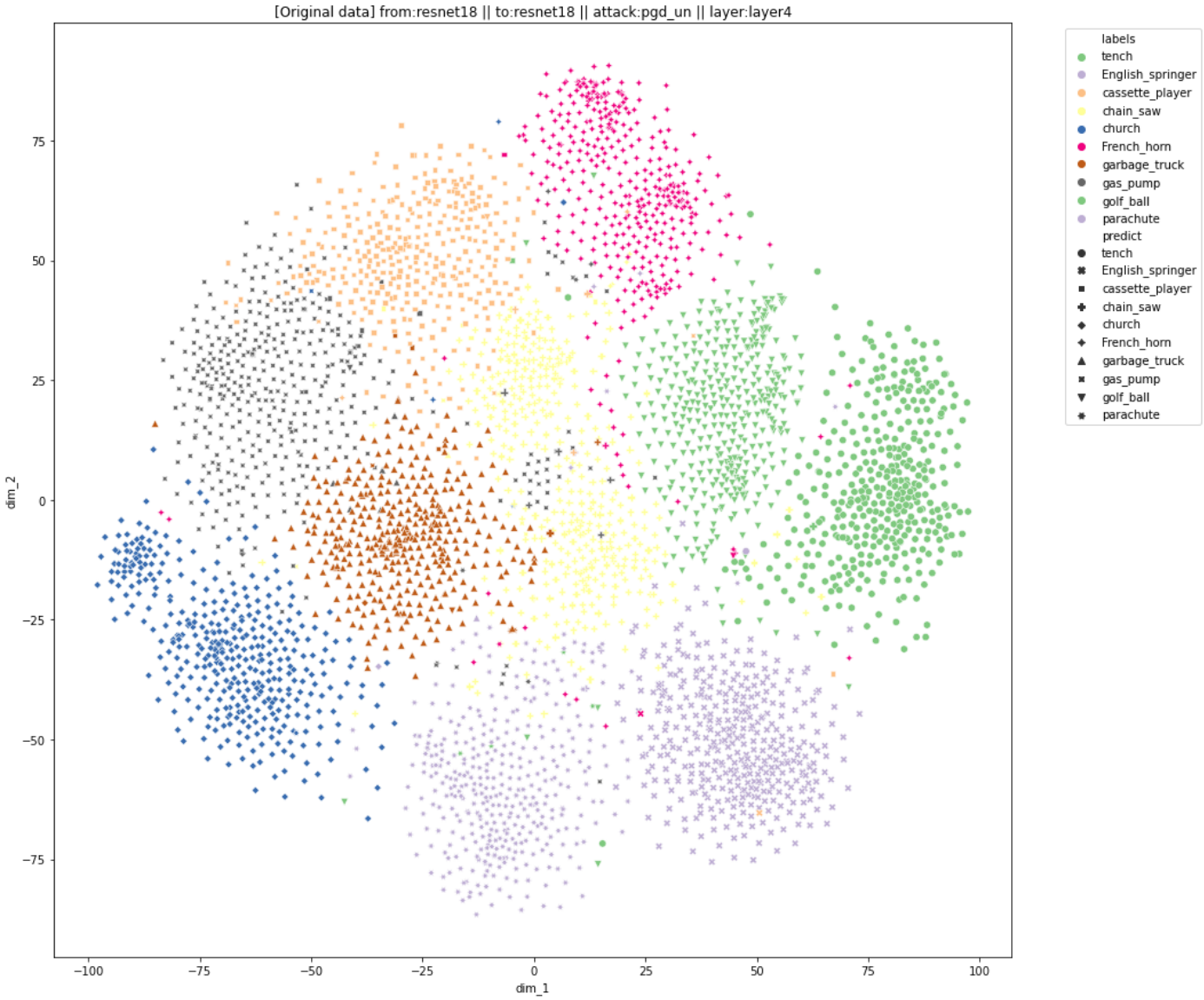

[Adversarial examples] from:resnet18 || to:resnet18 || attack:pgd_un || layer:layer1
labels
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
predict
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
dim_1
dim_2
100
50
0
−50
−100
−100
−50
0
50
100


[Original data] from:resnet18 || to:resnet18 || attack:pgd_un || layer:layer2
labels
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
predict
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
dim_1
dim_2
100
50
0
−50
−100
−100
−50
0
50
100


[Adversarial examples] from:resnet18 || to:resnet18 || attack:pgd_un || layer:layer2
dim_1
dim_2
labels
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
predict
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute


[Original data] from:resnet18 || to:resnet18 || attack:pgd_un || layer:layer3
dim_1
dim_2
100
50
0
−50
−100
−100
−50
0
50
100
labels
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
predict
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute


[Adversarial examples] from:resnet18 || to:resnet18 || attack:pgd_un || layer:layer3
dim_1
dim_2
labels
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
predict
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute


[Original data] from:resnet18 || to:resnet18 || attack:pgd_un || layer:layer4
dim_1
dim_2
labels
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
predict
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute

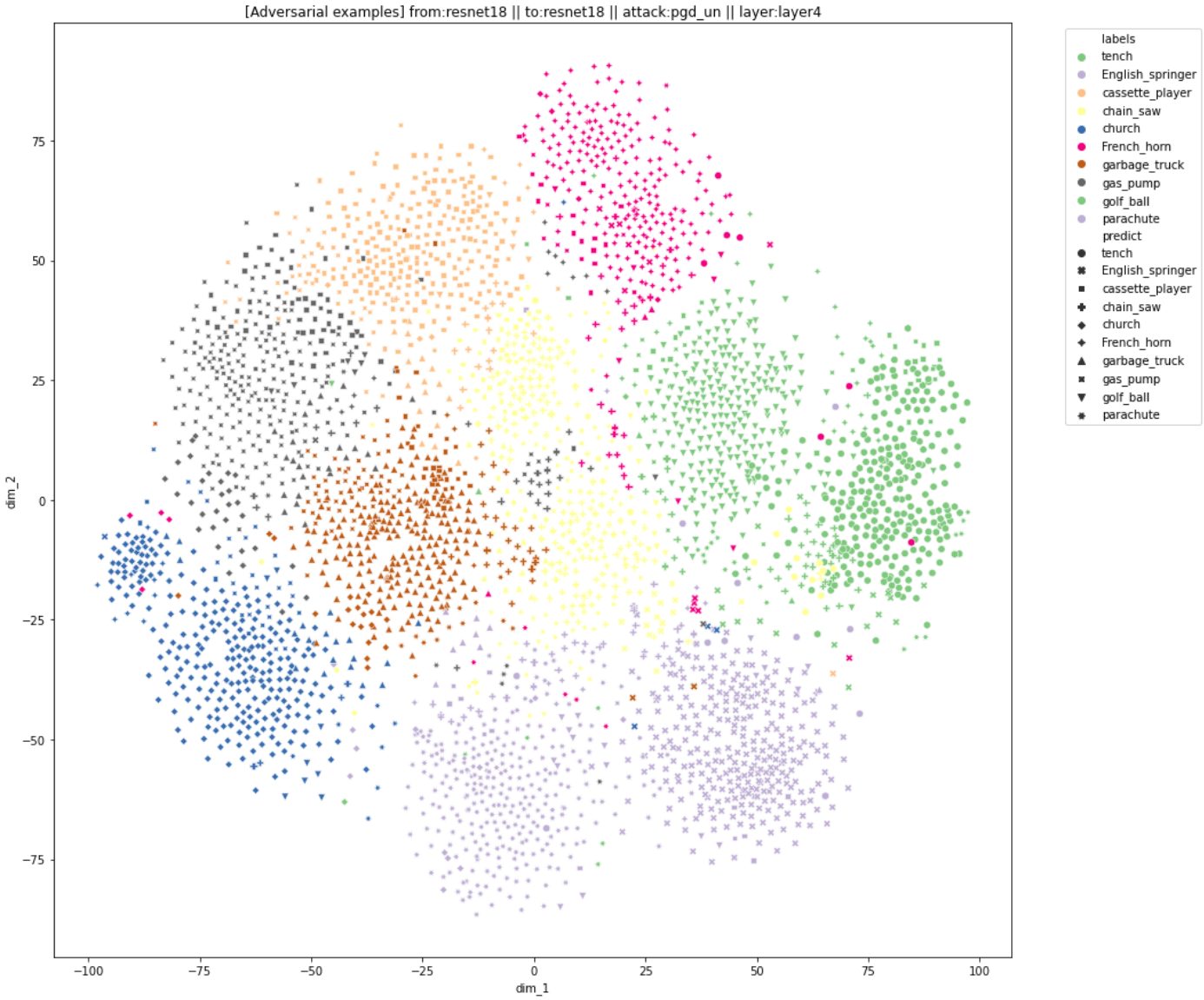
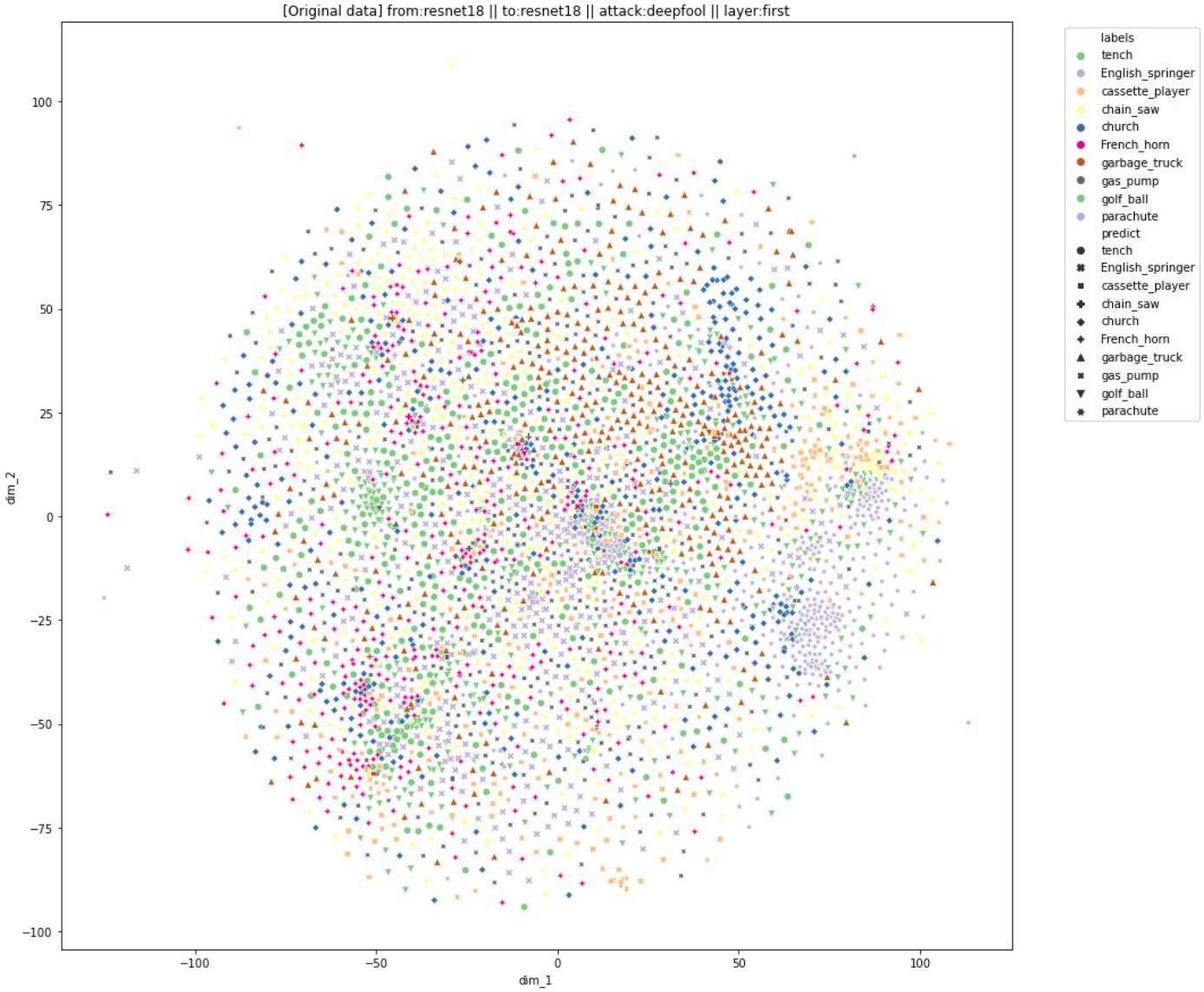
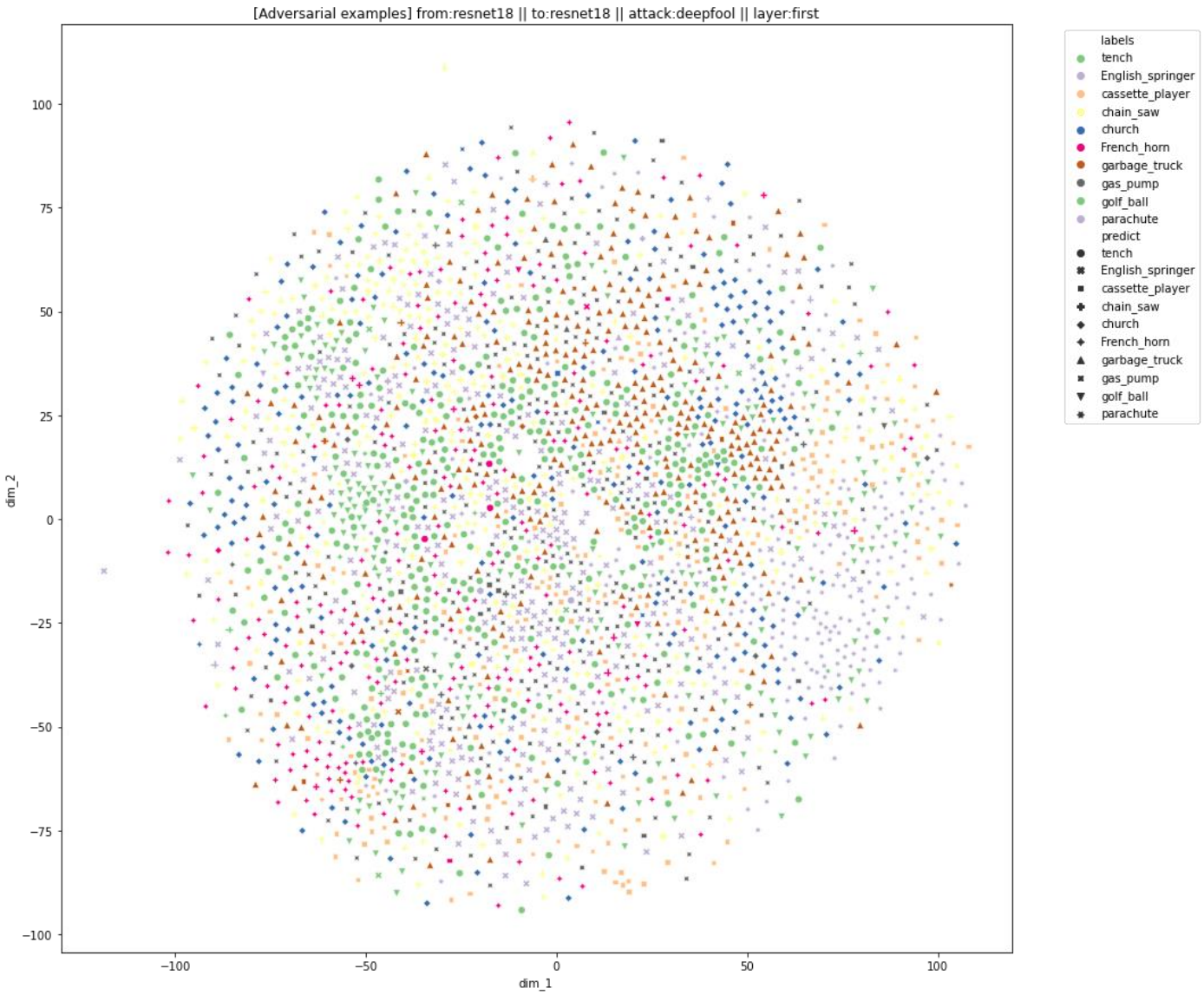
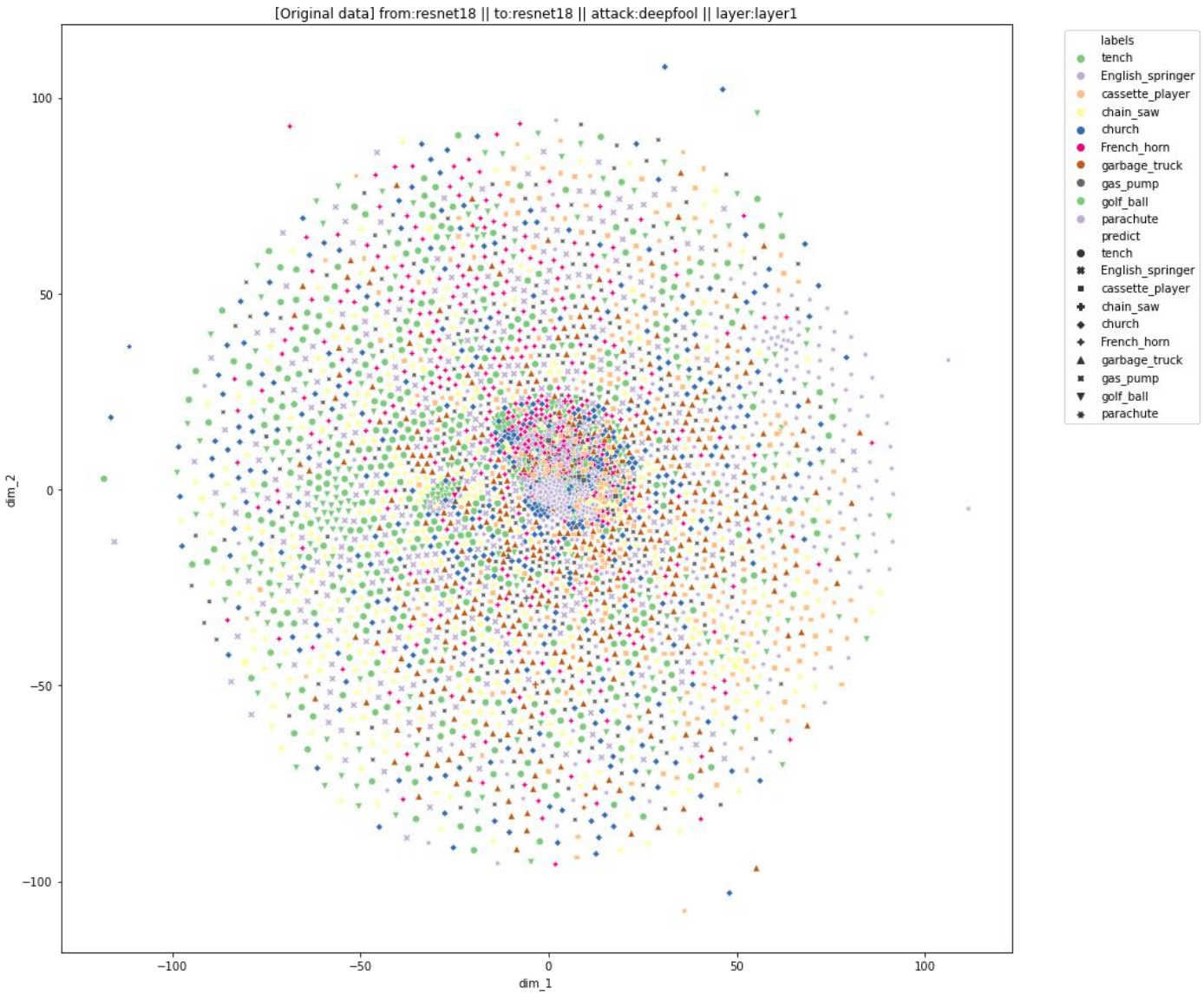


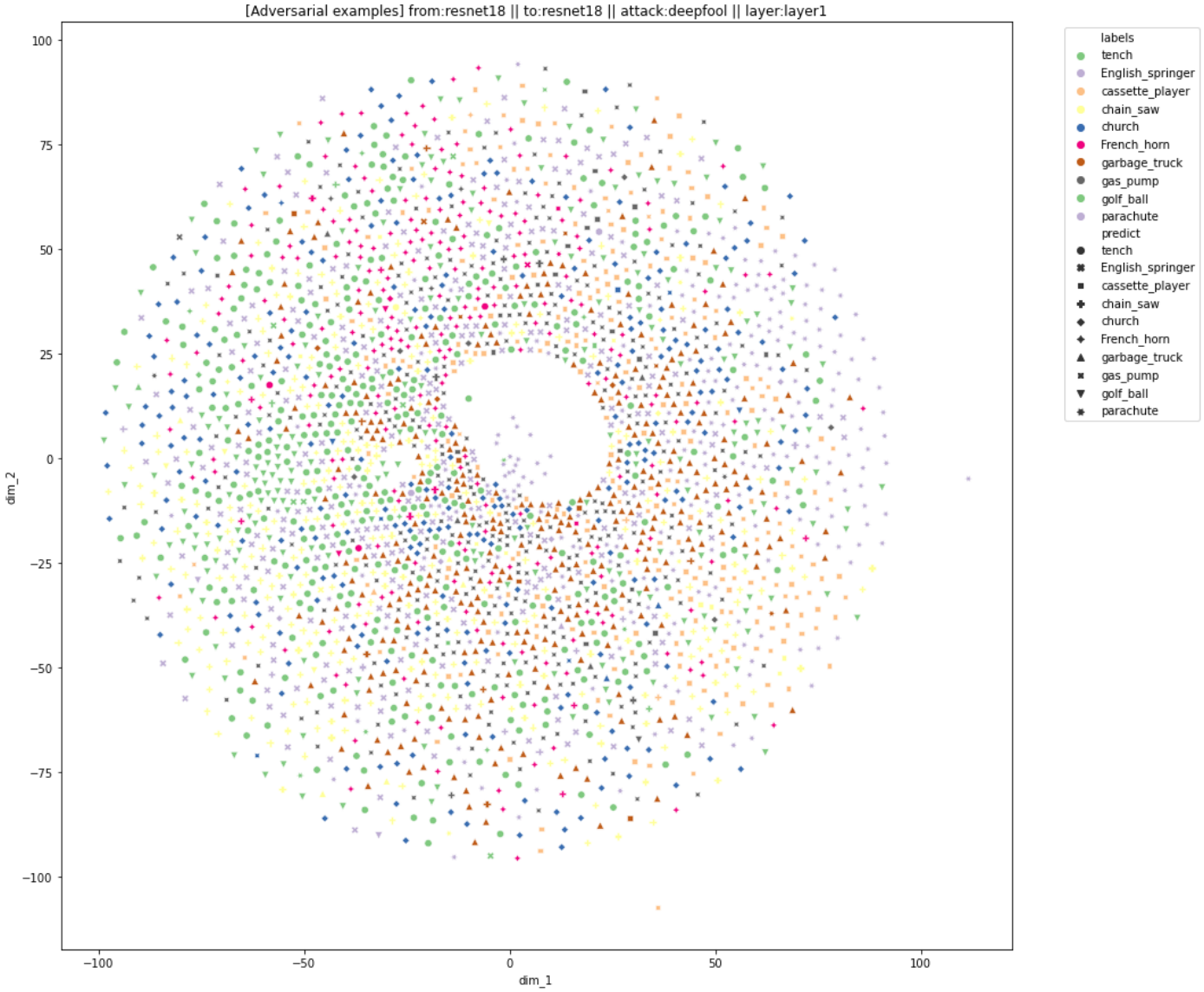
Figure 4. Embedding of adversarial examples generated by ResNet-18 with PGD algorithm from each layer of ResNet-18.

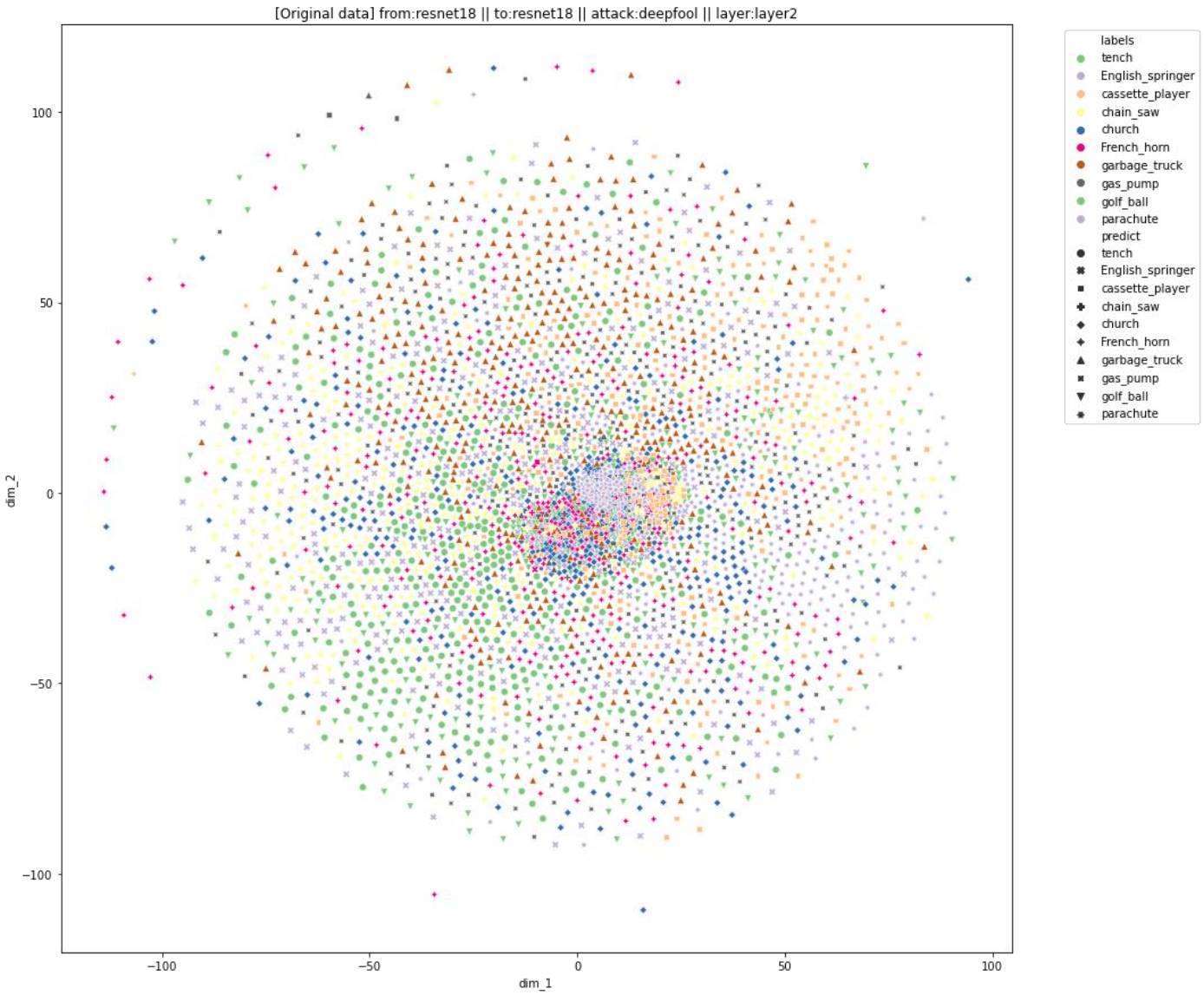
Considering the DeepFool algorithm, the behavior of the first, layer-1, layer-2, and layer-3 layers are also similar to the above case and visualized in Figure 5. It can generate the adversarial examples by leveraging critical decision boundary as well. However, the DeepFool has a fatal flaw that its attacking power will be lessened if the adversarial examples are saved with “int8” format, which round the floating point in the adversarial examples and make the attack less successful.


[Original data] from:resnet18 || to:resnet18 || attack:deepfool || layer:first
dim_1
dim_2
labels
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
predict
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute

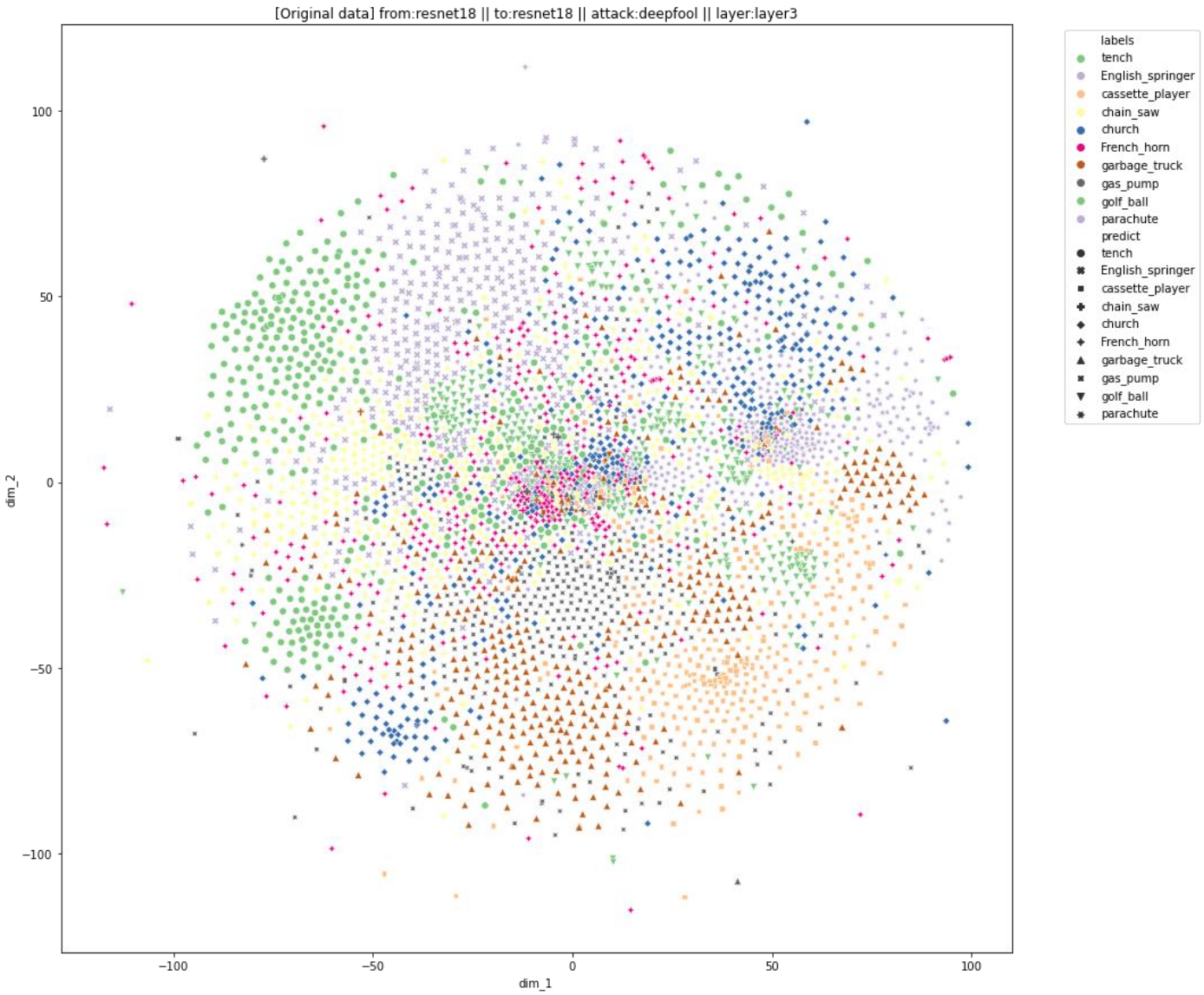

[Adversarial examples] from:resnet18 || to:resnet18 || attack:deepfool || layer:first
dim_1
dim_2
labels
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
predict
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute

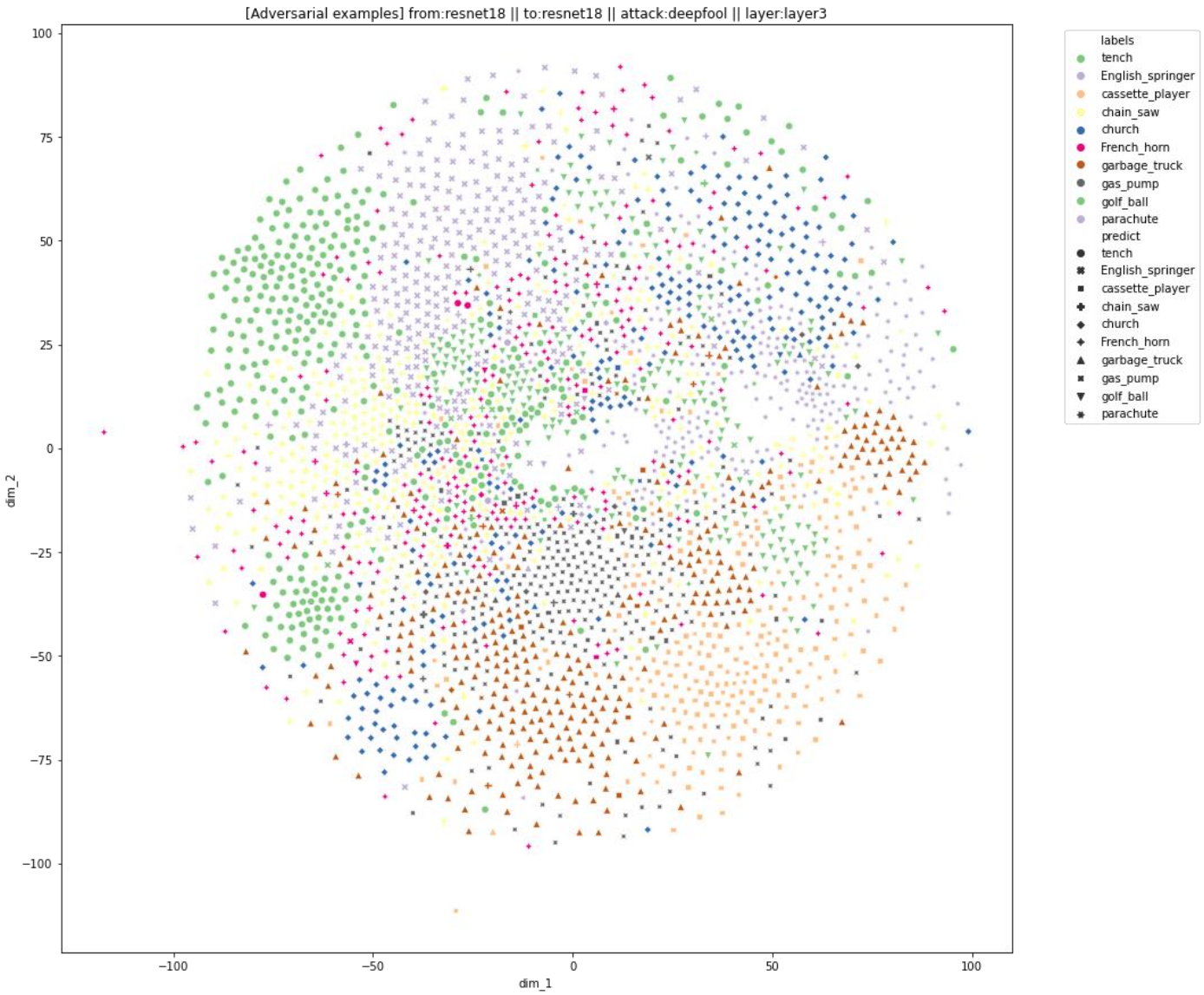

[Original data] from:resnet18 || to:resnet18 || attack:deepfool || layer:layer1
labels
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
predict
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
dim_1
dim_2
100
50
0
−50
−100
−100
−50
0
50
100

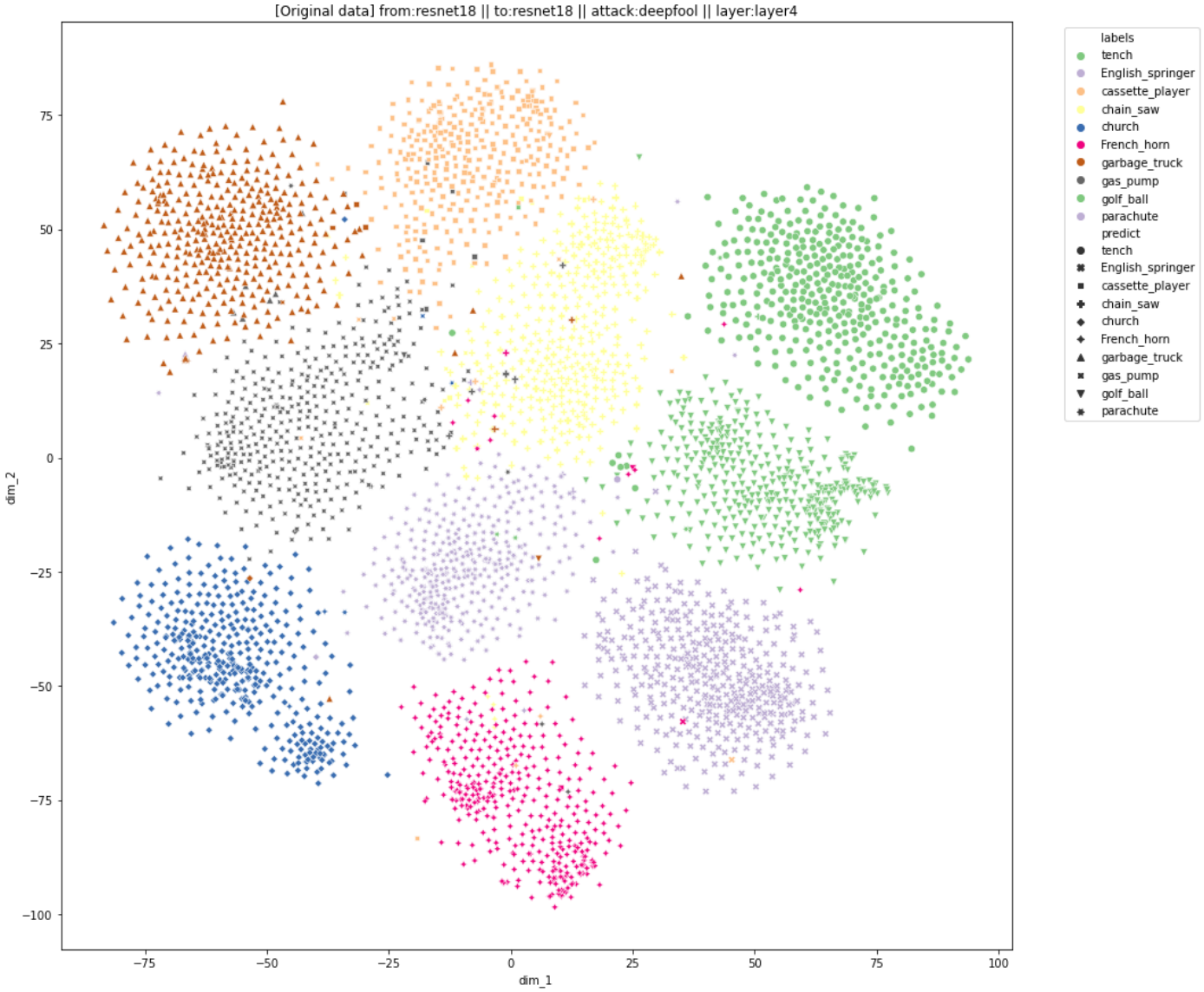

[Adversarial examples] from:resnet18 || to:resnet18 || attack:deepfool || layer:layer1
dim_1
dim_2
labels
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
predict
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute


[Original data] from:resnet18 || to:resnet18 || attack:deepfool || layer:layer2
dim_1
dim_2
100
50
0
−50
−100
−100
−50
0
50
100
labels
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
predict
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute


[Adversarial examples] from:resnet18 || to:resnet18 || attack:deepfool || layer:layer2
dim_1
dim_2
labels
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
predict
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute


[Original data] from:resnet18 || to:resnet18 || attack:deepfool || layer:layer3
dim_1
dim_2
100
50
0
−50
−100
−100
−50
0
50
100
labels
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
predict
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute


[Adversarial examples] from:resnet18 || to:resnet18 || attack:deepfool || layer:layer3
dim_1
dim_2
labels
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
predict
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute


[Original data] from:resnet18 || to:resnet18 || attack:deepfool || layer:layer4
dim_1
dim_2
labels
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
predict
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute

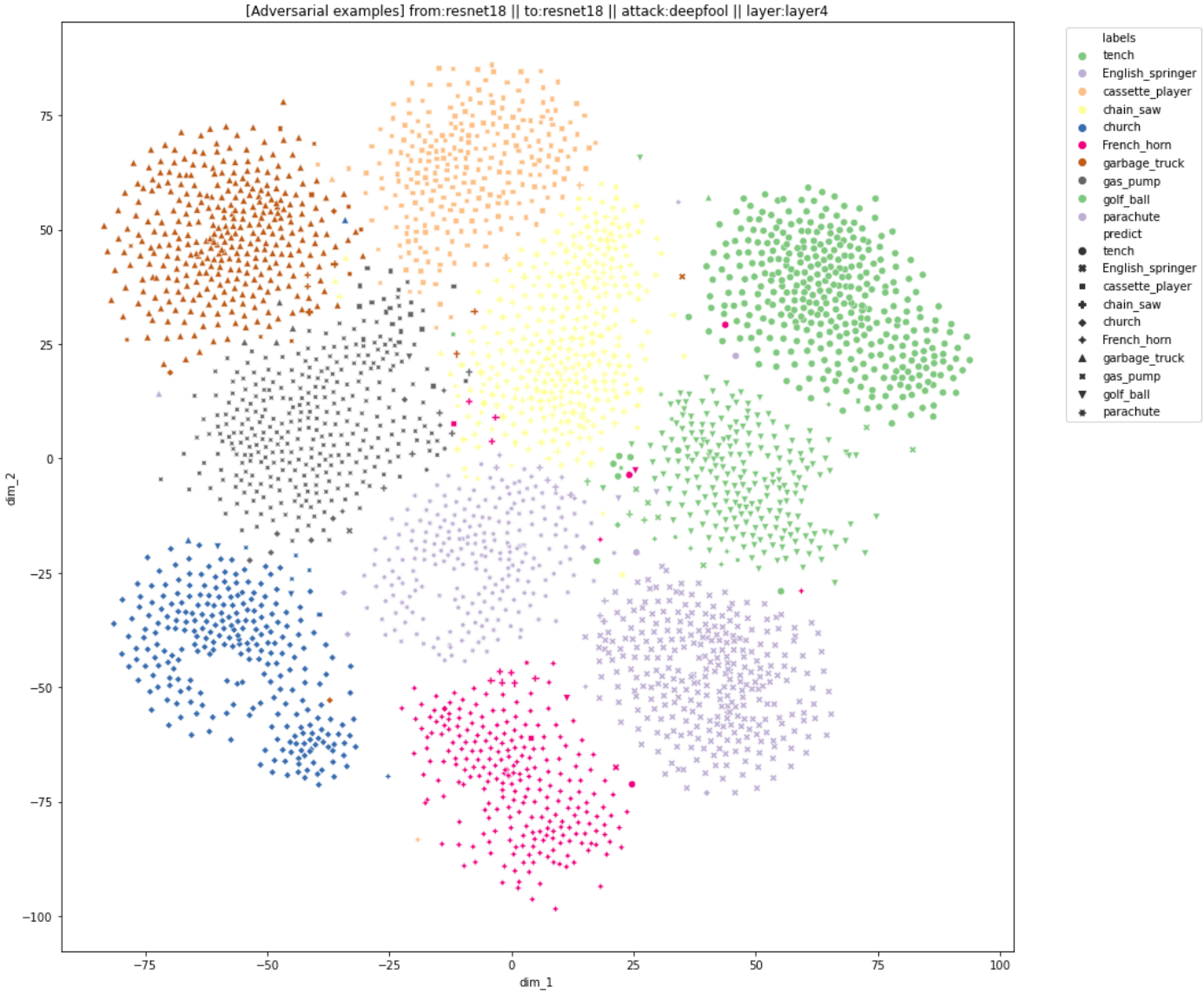
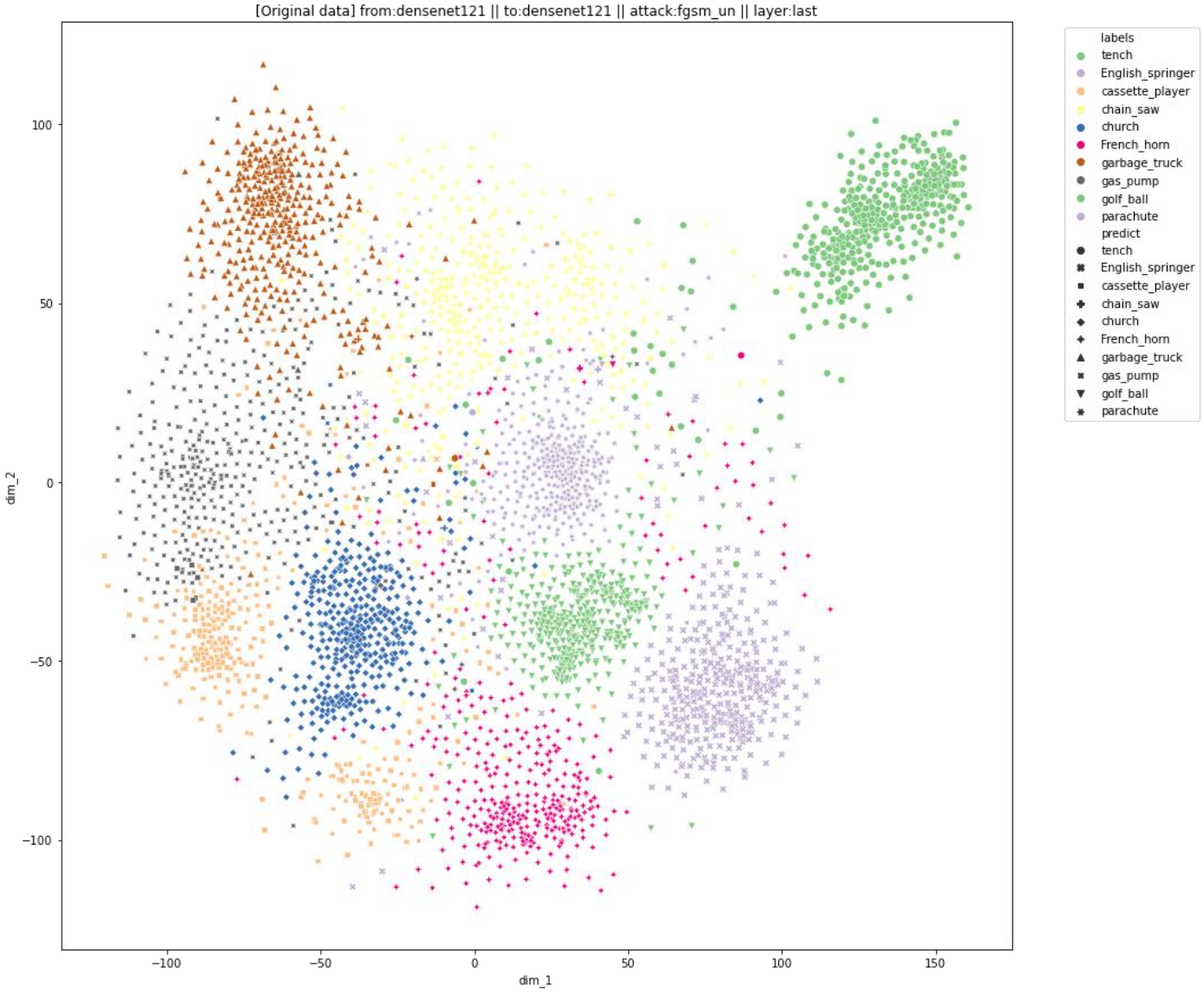


Figure 5. Embedding of adversarial examples generated by ResNet-18 with DeepFool algorithm from each layer of ResNet-18.

Lastly, the case of adversarial examples generated by DenseNet -121 via FGSM against DenseNet-121 is shown in Figure 6. In this case, it can be observed that the FGSM also has the ability to find the critical decision boundary to make the model erroneously predict the results.


[Original data] from:densenet121 || to:densenet121 || attack:fgsm_un || layer:last
labels
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
predict
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
dim_1
dim_2
100
50
0
−50
−100
−100
−50
0
50
100
150

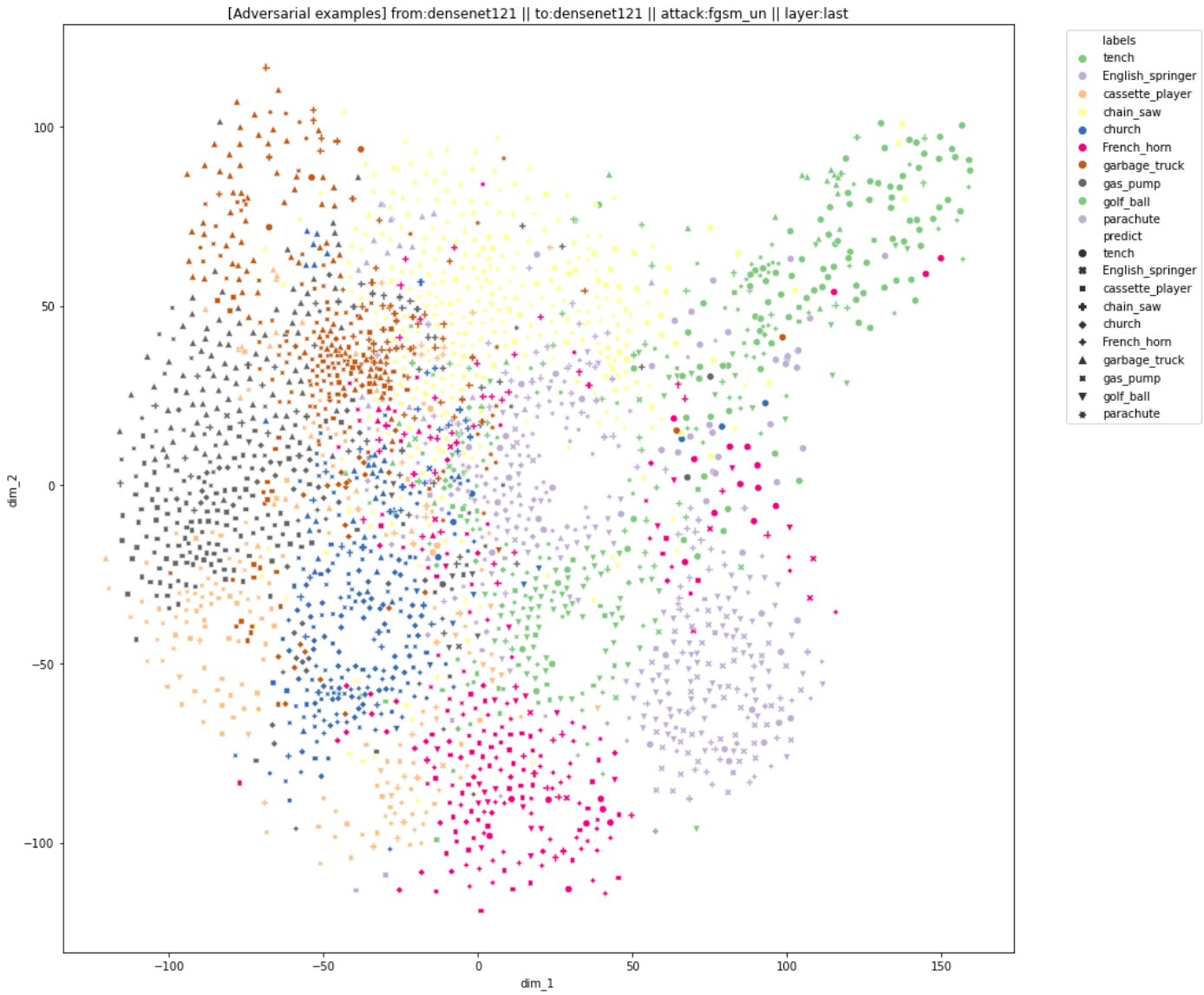
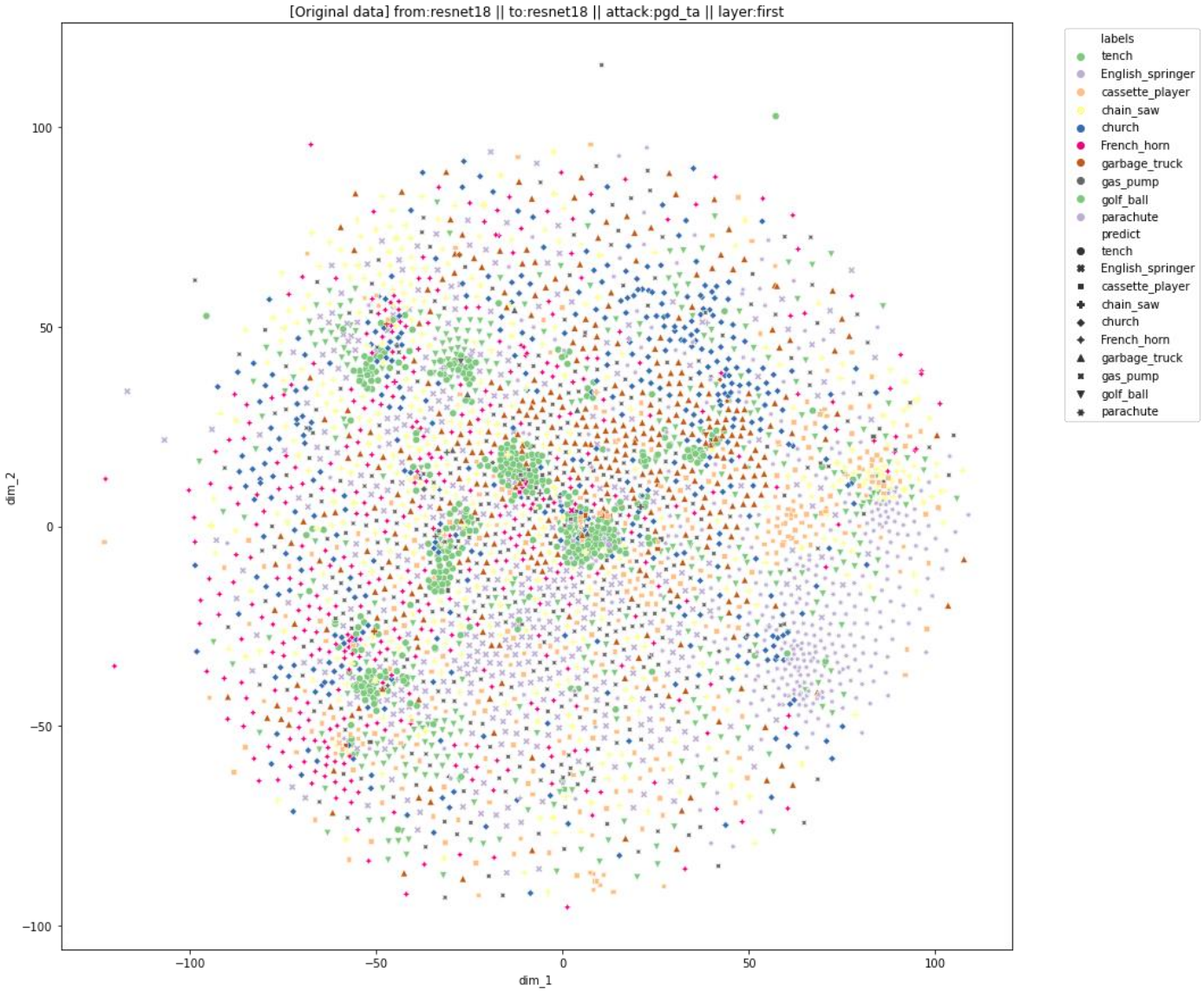
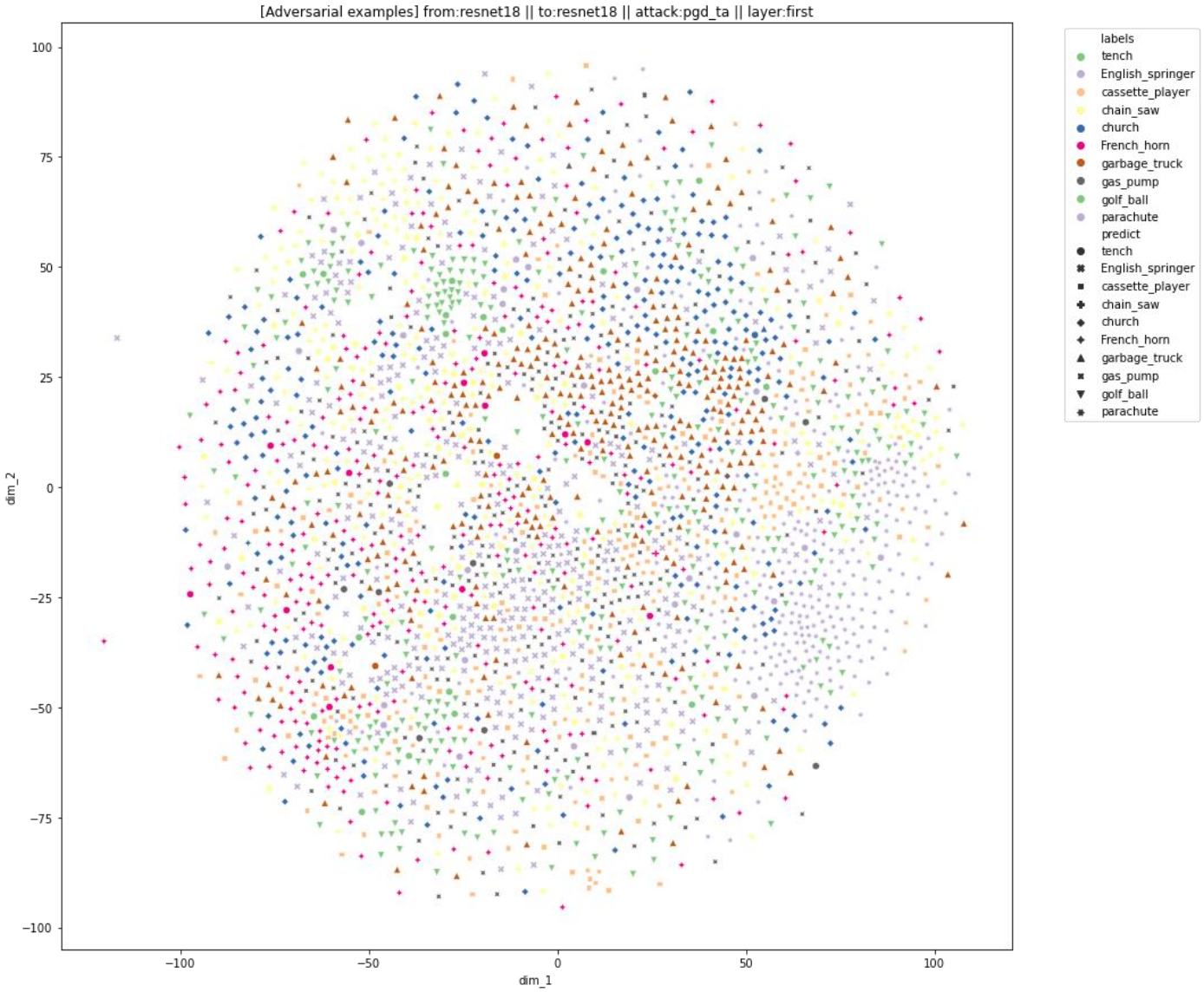


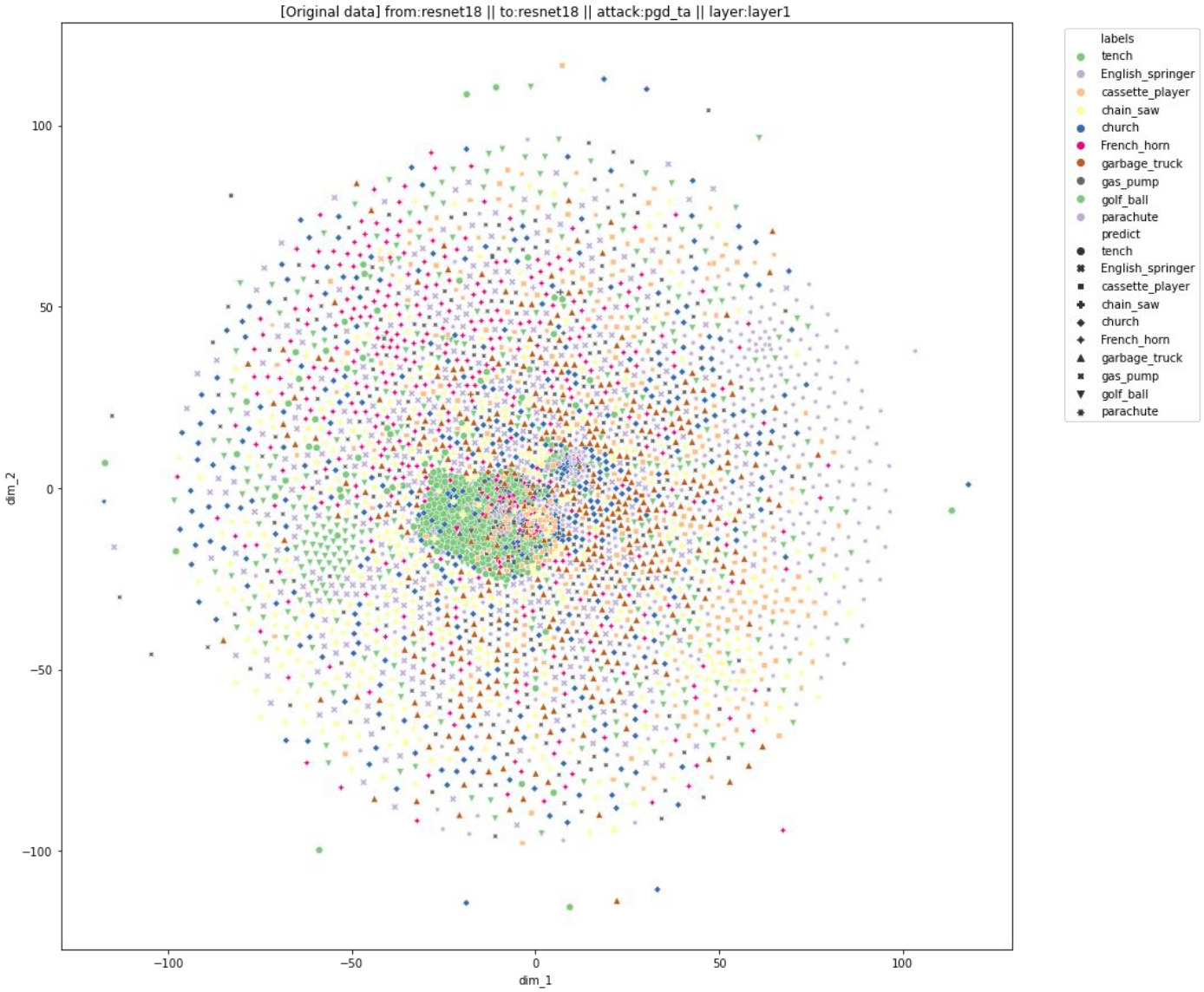
Figure 6. Embedding of adversarial examples generated by DenseNet-121 with FGSM algorithm from each layer of DenseNet-121.

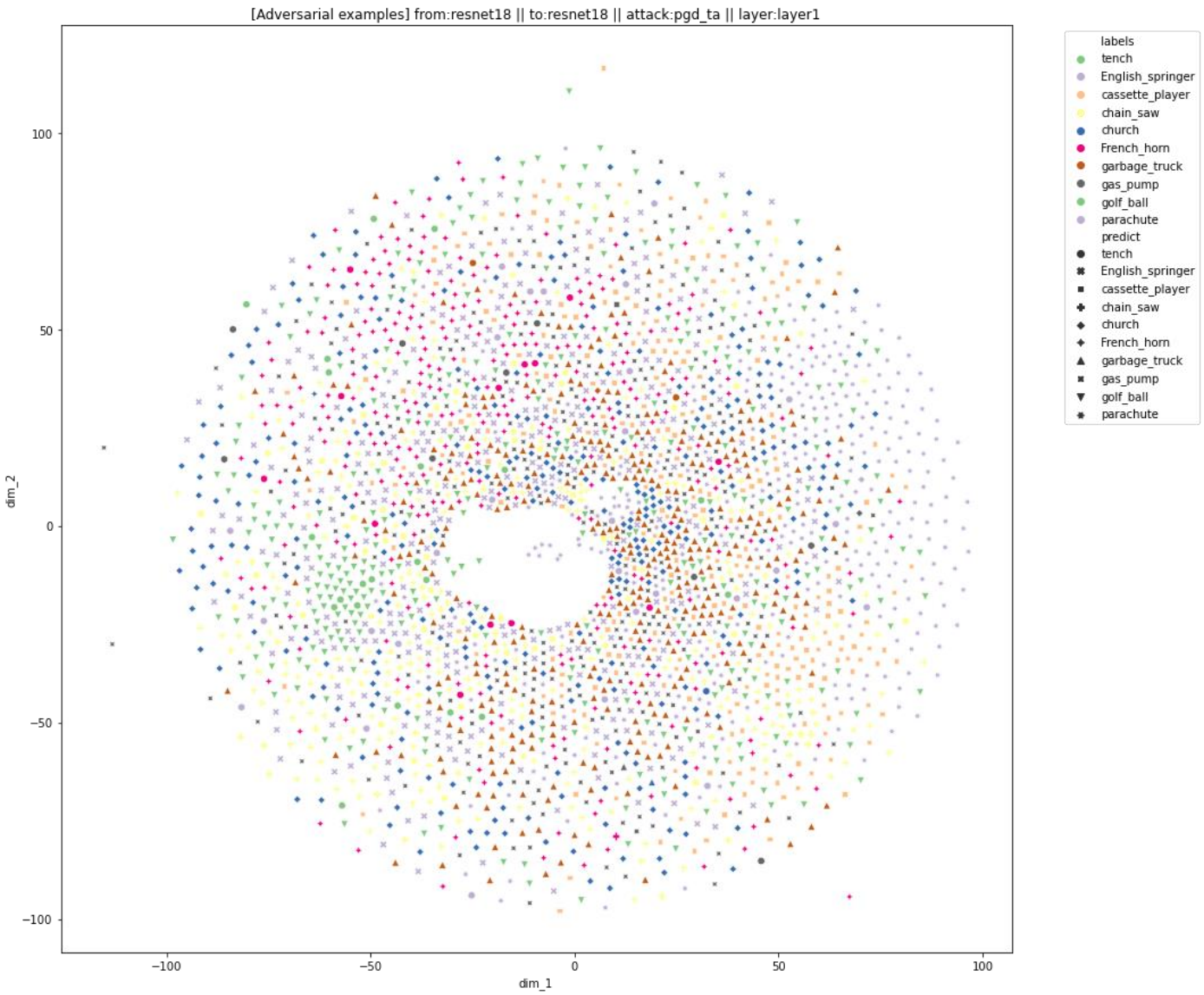
3.6. Embedding analysis of targeted attack

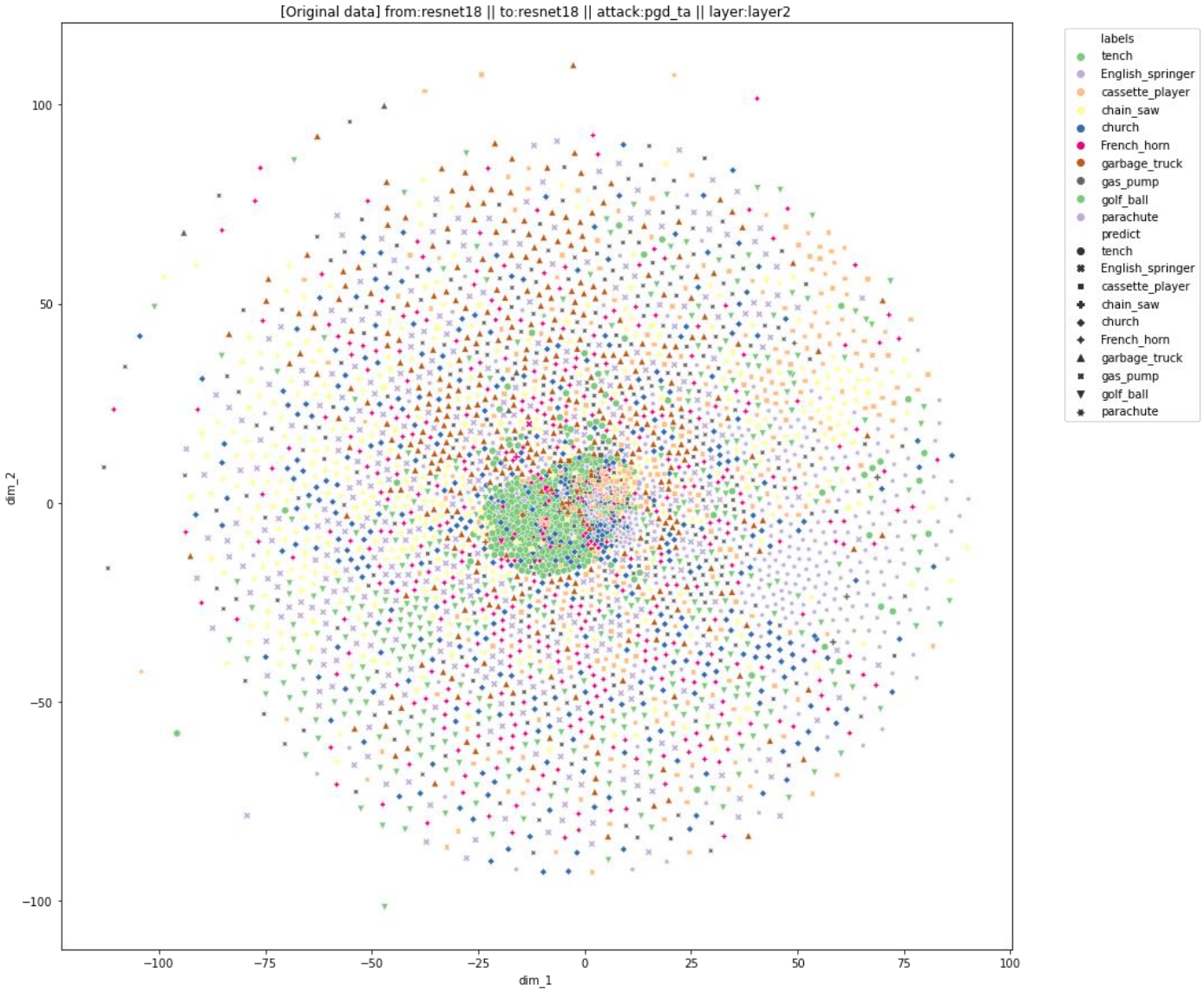
Figure 7 visualizes the scenario of targeted attack that attempts to make all the adversarial examples predicted as tench class performed by adversarial examples generated by ResNet-18 via PGD algorithm against ResNet-18. The behavior in the first, layer-1, layer-2, and layer-3 layers are similar. However, the attacking power is also lessened by the problem of “int8” rounding. In the layer-4, it is obviously that there are several critical points that can be classified as tench although these points are close to corresponding clustering center. This experiment also demonstrates the vulnerability might come from the layer-4 and the fully-connected layer.

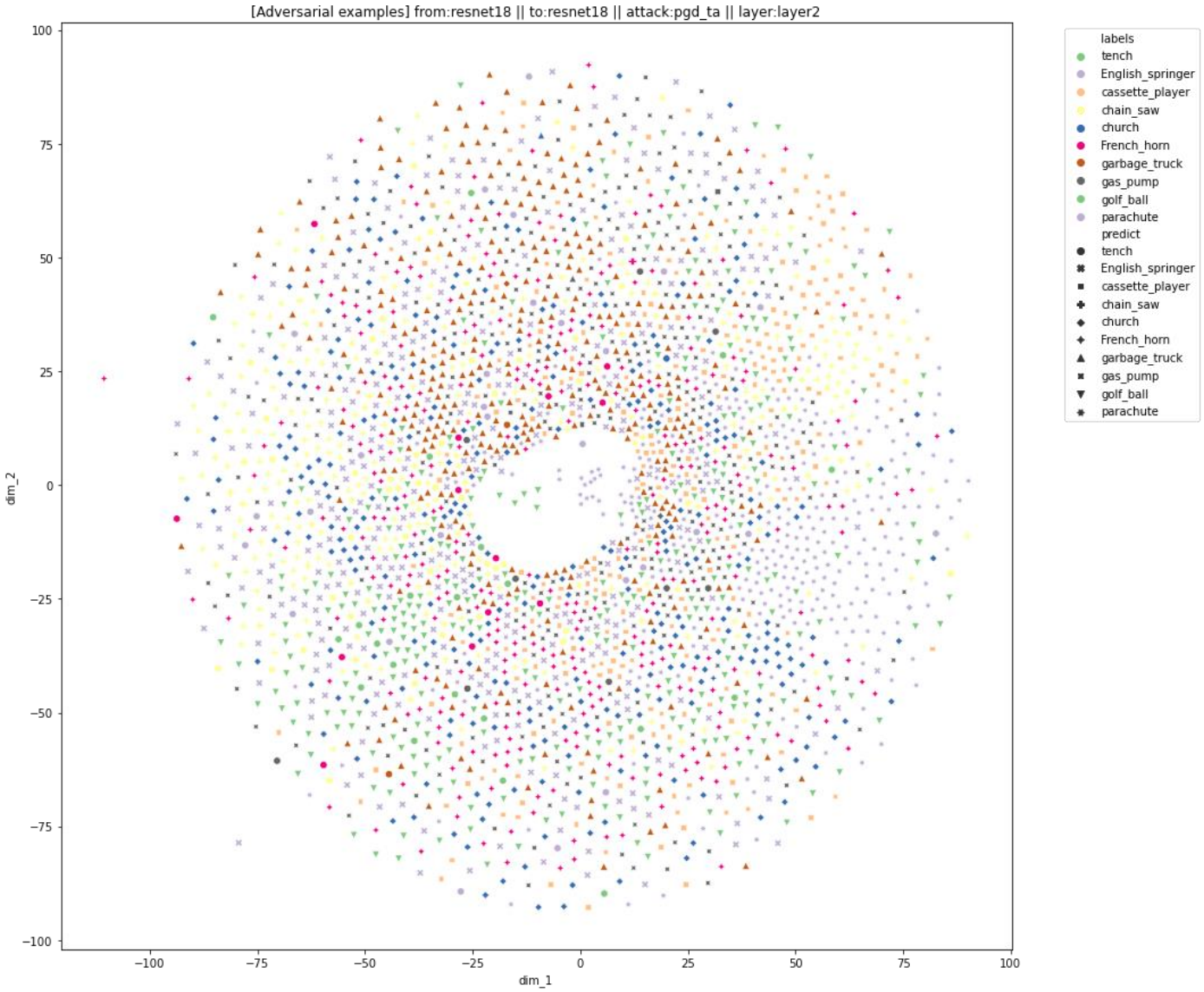

[Original data] from:resnet18 || to:resnet18 || attack:pgd_ta || layer:first
dim_1
dim_2
100
50
0
−50
−100
−100
−50
0
50
100
labels
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
predict
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute

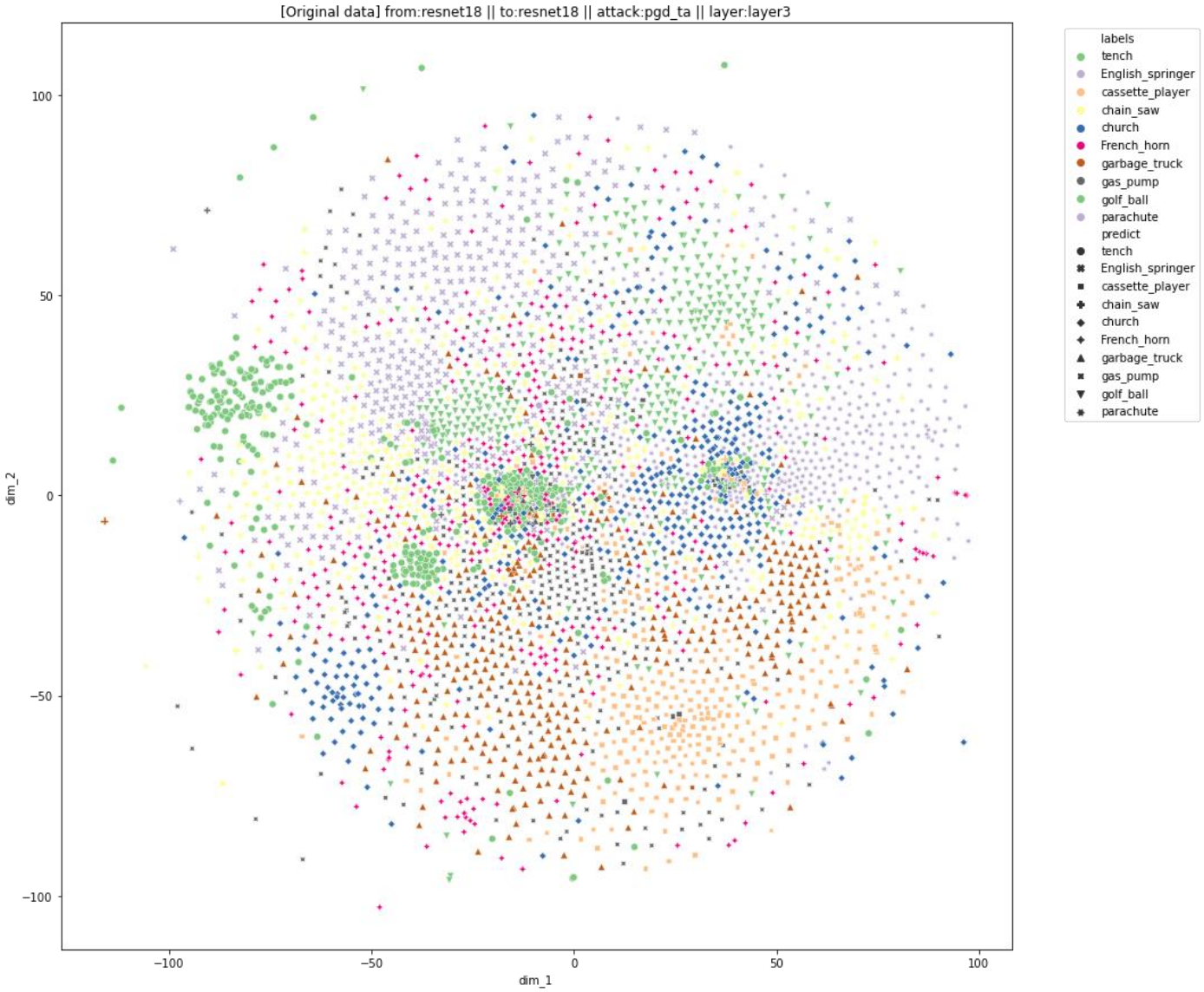

[Adversarial examples] from:resnet18 || to:resnet18 || attack:pgd_ta || layer:first
dim_1
dim_2
labels
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
predict
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute

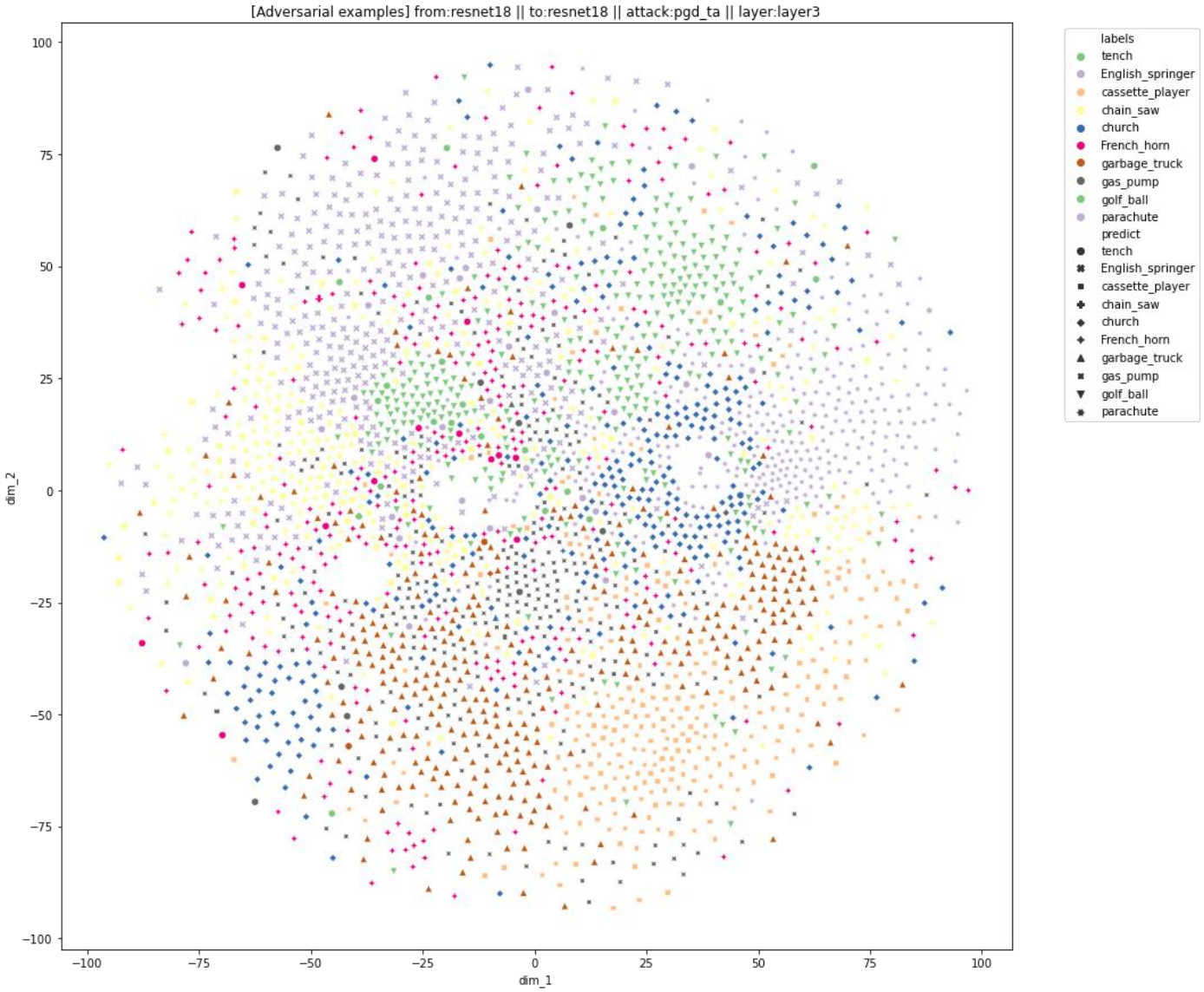

[Original data] from:resnet18 || to:resnet18 || attack:pgd_ta || layer:layer1
labels
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
predict
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
dim_1
dim_2
100
50
0
−50
−100
−100
−50
0
50
100

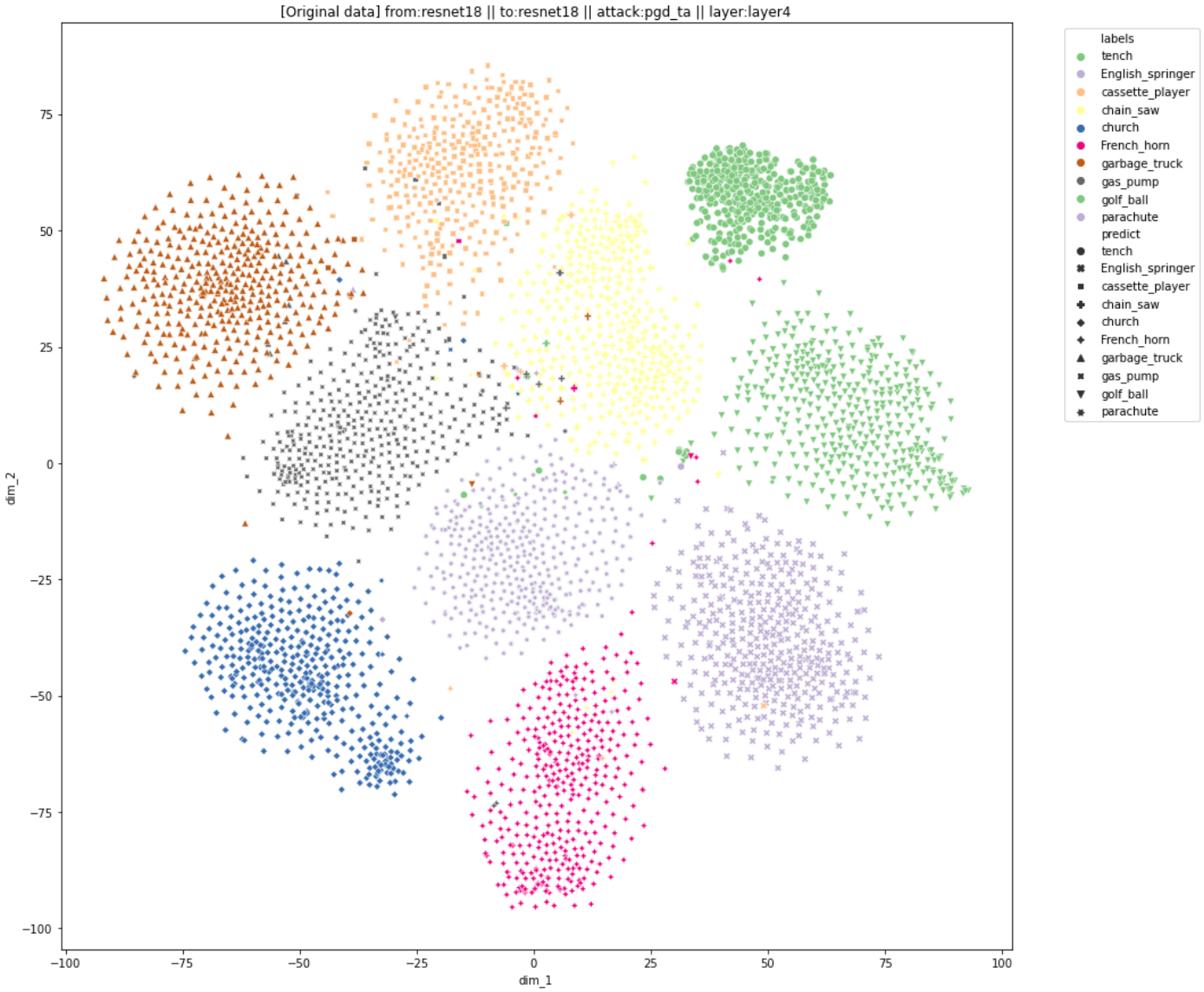

[Adversarial examples] from:resnet18 || to:resnet18 || attack:pgd_ta || layer:layer1
dim_1
dim_2
100
50
0
−50
−100
−100
−50
0
50
100
labels
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
predict
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute


[Original data] from:resnet18 || to:resnet18 || attack:pgd_ta || layer:layer2
labels
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
predict
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
dim_1
dim_2
100
50
0
−50
−100
−100
−75
−50
−25
0
25
50
75
100


[Adversarial examples] from:resnet18 || to:resnet18 || attack:pgd_ta || layer:layer2
dim_1
dim_2
labels
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
predict
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute


[Original data] from:resnet18 || to:resnet18 || attack:pgd_ta || layer:layer3
labels
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
predict
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
dim_1
dim_2
100
50
0
−50
−100
−100
−50
0
50
100


[Adversarial examples] from:resnet18 || to:resnet18 || attack:pgd_ta || layer:layer3
dim_1
dim_2
labels
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
predict
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute


[Original data] from:resnet18 || to:resnet18 || attack:pgd_ta || layer:layer4
dim_1
dim_2
labels
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute
predict
tench
English_springer
cassette_player
chain_saw
church
French_horn
garbage_truck
gas_pump
golf_ball
parachute

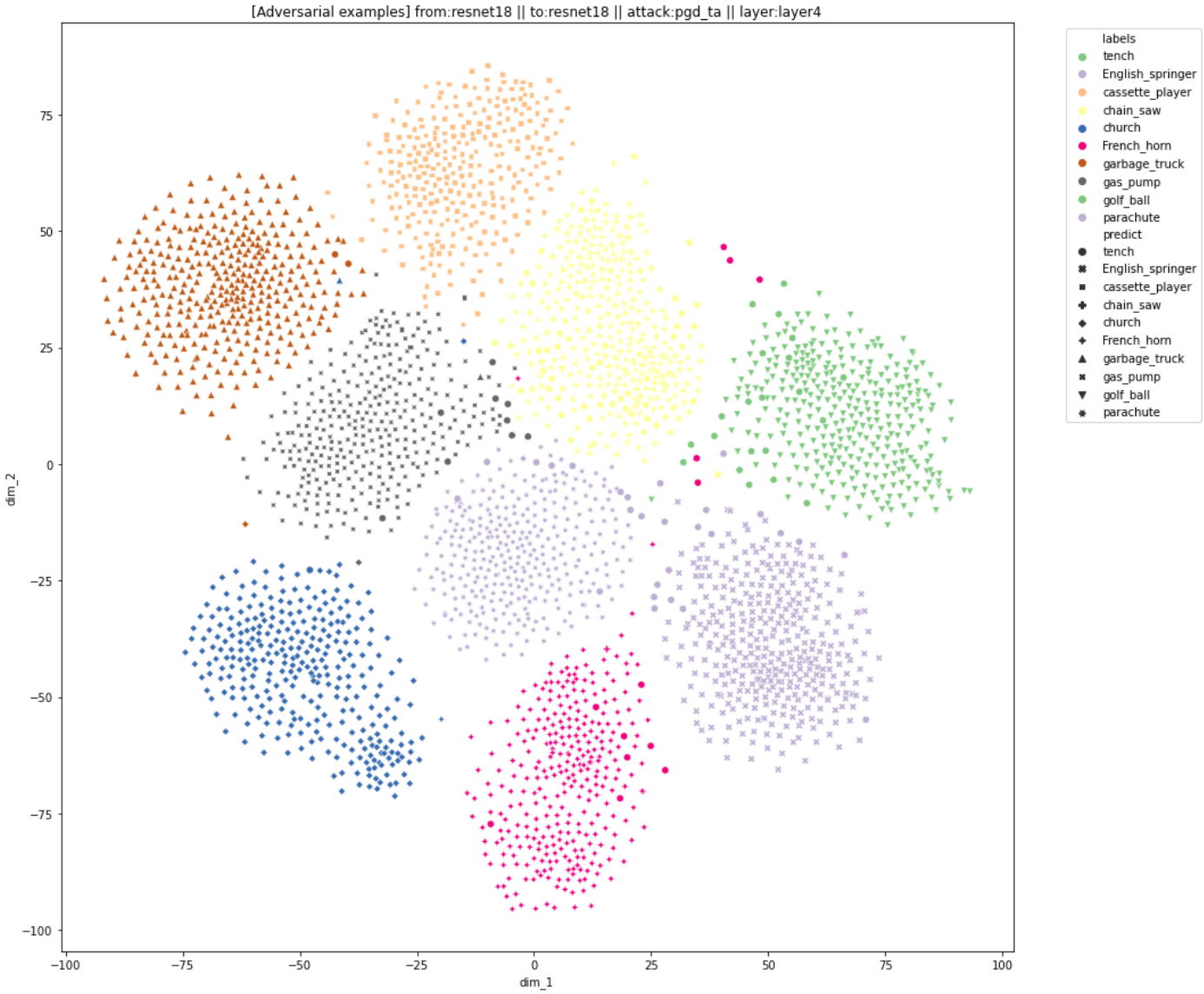


Figure 7. Embedding of adversarial examples generated by ResNet-18 with PGD algorithm from each layer of ResNet-18 to perform targeted attack.

4. Conclusion

With the complete experiments in this study, it can be supported that the vulnerability of the deep model come from that last few layers that corresponding to high level features processing but not the first and middle layers which might probably be in charge of simple features capturing.

Also, by the embedding visualization, we demonstrated that there are several critical decision boundaries that can be found and make the model misclassify the adversarial examples. These decision boundaries are really close to the original data and should be considered in the training process, or it might really threaten the model because the small perturbation will cause model to predict erroneously.

There are several limitations in this study. First, the iterative-based algorithms that generates the adversarial examples are too time-consuming and cannot be fully explored. Second, the problem of “int8” should be further solved either by directly treated with the floating point or by enlarge the perturbation limitation. Third, the small subset of ImageNet would probably not to generalize to whole 1000 classes of ImageNet.

However, this is still the first study that investigates the embedding from each layer and obtains a trustful reason why a deep learning model is vulnerable to adversarial attack. For more explication and explanation, further investigation should be performed to reveal the mystery of deep learning model.

5. Reference

[1] Grewal, M., Srivastava, M. M., Kumar, P., & Varadarajan, S. (2018, April). Radnet: Radiologist level accuracy using deep learning for hemorrhage detection in ct scans. In 2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018) (pp. 281-284). IEEE.

[2] Shafique, S., & Tehsin, S. (2018). Acute lymphoblastic leukemia detection and classification of its subtypes using pretrained deep convolutional neural networks. Technology in cancer research & treatment, 17, 1533033818802789.

[3] Hermsen, M., de Bel, T., Den Boer, M., Steenbergen, E. J., Kers, J., Florquin, S., ... & van der Laak, J. A. (2019). Deep learning–based histopathologic assessment of kidney tissue. Journal of the American Society of Nephrology, 30(10), 1968-1979.

[4] Yuan, X., He, P., Zhu, Q., & Li, X. (2019). Adversarial examples: Attacks and defenses for deep learning. IEEE transactions on neural networks and learning systems, 30(9), 2805-2824.

[5] Nugraha, B. T., & Su, S. F. (2017, October). Towards self-driving car using convolutional neural network and road lane detector. In 2017 2nd International Conference on Automation, Cognitive Science, Optics, Micro Electro-Mechanical System, and Information Technology (ICACOMIT) (pp. 65-69). IEEE.

[6] Goodfellow, I. J., Shlens, J., & Szegedy, C. (2014). Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572.

[7] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083.

[8] Dong, Y., Liao, F., Pang, T., Su, H., Zhu, J., Hu, X., & Li, J. (2018). Boosting adversarial attacks with momentum. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 9185-9193).

[9] Moosavi-Dezfooli, S. M., Fawzi, A., & Frossard, P. (2016). Deepfool: a simple and accurate method to fool deep neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2574-2582).

[10] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[11] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).

[12] Zagoruyko, S., & Komodakis, N. (2016). Wide residual networks. arXiv preprint arXiv:1605.07146.

[13] Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. Journal of machine learning research, 9(11).

[14] https://github.com/fastai/imagenette