CACHING IN DNNS - SPEEDING UP INFERENCE FOR SIMILAR INPUTS

Shrisudhan
me17b122@smail.iitm.ac.in

SysDL Project
CS 6886

Problem definition

- Speed up the inference time for an input which is very similar to another pre-executed input by using caching.
- Defining a function which maps the input to the intermediate state which can be compared withdata stored in cache memory, called hashing function.
- Hash map should be so designed that it maps inputs which belong to the same class to have similar hash values.
- Different DL networks will be taken and caches from those networks will be obtained. Few learnable hashing functions will be trained to fit the cache value and cache key and inference speed-up will be compared.

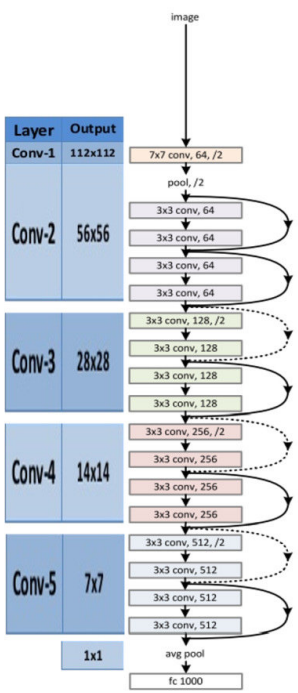
Related works

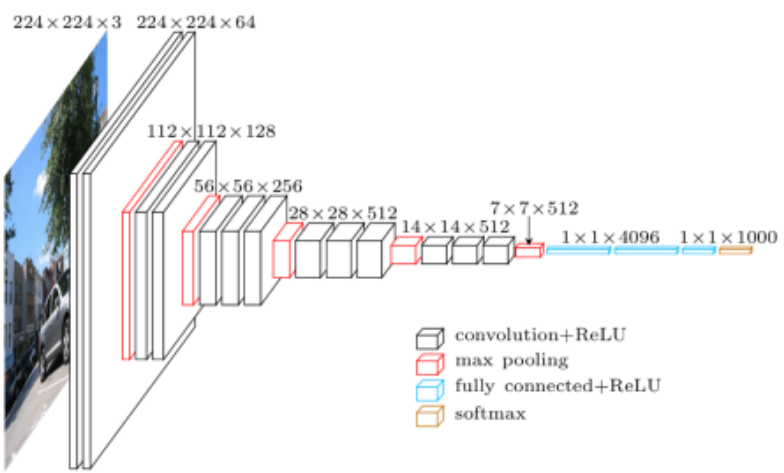
- A Simple Cache Model for Image Recognition
  https://arxiv.org/pdf/1805.08709.pdf
- Deep Hashing: A Joint Approach for Image Signature Learning
  https://arxiv.org/pdf/1608.03658.pdf
- FaceNet: A Unified Embedding for Face Recognition and Clustering
  https://arxiv.org/pdf/1503.03832.pdf
- Learnable Histogram: Statistical Context Features for Deep Neural Networks
  https://arxiv.org/pdf/1804.09398.pdf
- Deep Fisher Networks for Large-Scale Image Classification
  https://papers.nips.cc/paper/4926-deep-fisher-networks-for-large-scale-image-classification.pdf
- Improved Deep Metric Learning with Multi-class N-pair Loss Objective
  http://www.nec-labs.com/uploads/images/Department-Images/MediaAnalytics/papers/nips16_npairmetriclearning.pdf

Key technical insights

Image classification : ResNet-18 and VGG-16

- In this project the two most famous image classification networks are used, namely ResNet-18 and VGG-19.
- These models are used for generating cache and the improvement in inference time is also measured with reference to these models.

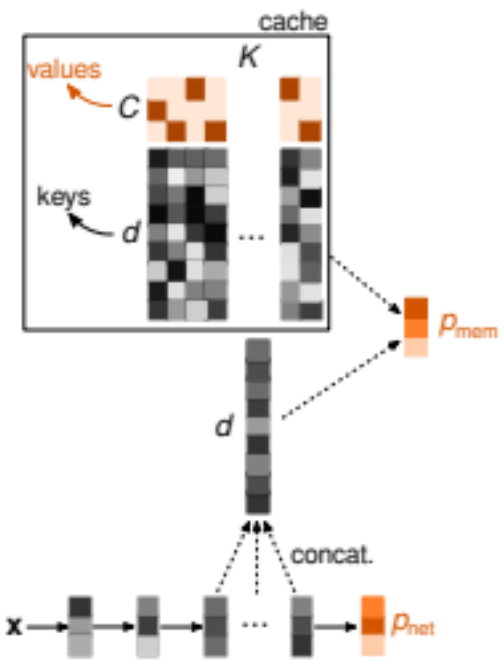




Key technical insights

Caching technique : A Simple Cache Model for Image Recognition

- Cache memory is stored with cache keys and cache values.
- Cache keys are of dimension dxK where d is the dimensionality of the cache vector and K is the number of items stored in cache memory.
- The key vector,μ_K for a particular item x_K in the training set is obtained by taking the activities of one or more layers of the network when x_K is input to a deep convolutional network, and concatenating them.
- The value vector,ν_K is just the class representation.



Key technical insights

Given a test item x with label y (considered as a one-hot vector), its similarity with the stored items in the cache is computed as follows:

$$\sigma_k(x) = e^{(\theta\phi(x)^T \mu_k)}$$

$\phi(x)$ => cache computed for input x

$$p_{\text{mem}}(\mathbf{y}|\mathbf{x}) = \frac{\sum_{k=1}^{K} \upsilon_k \sigma_k(\mathbf{x})}{\sum_{k=1}^{K} \sigma_k(\mathbf{x})}$$

The actual predicted distribution of the network is given as from the combination of the prediction from cache and from the original network.

$$p(y|x) = (1-\lambda)p_{net}(y|x) + \lambda p_{mem}(y|x)$$

The two hyper-parameters to be tuned are λ and θ.

Key technical insights

Results:

Cache performance on various networks

Model	Params	C-10+	C-100+	ImageNet
ResNet20 ($\lambda = 0$)	0.27M	8.33	32.66	–
ResNet20-Cache3	0.27M	**7.58**	**29.18**	–
ResNet20-Cache3-CacheOnly ($\lambda = 1$)	0.27M	11.87	39.18	–
ResNet32 ($\lambda = 0$)	0.46M	7.74	32.94	–
ResNet32-Cache3	0.46M	**7.01**	**29.36**	–
ResNet32-Cache3-CacheOnly ($\lambda = 1$)	0.46M	11.13	38.40	–
ResNet56 ($\lambda = 0$)	0.85M	8.74	31.11	–
ResNet56-Cache3	0.85M	**8.11**	**27.99**	–
ResNet56-Cache3-CacheOnly ($\lambda = 1$)	0.85M	13.05	34.06	–
DenseNet40 ($\lambda = 0$)	1M	5.75	27.08	–
DenseNet40-Cache2	1M	**5.44**	**25.25**	–
DenseNet40-Cache2-CacheOnly ($\lambda = 1$)	1M	8.68	37.64	–
DenseNet100 ($\lambda = 0$)	7M	5.08	22.62	–
DenseNet100-Cache1	7M	**4.92**	**21.95**	–
DenseNet100-Cache1-CacheOnly ($\lambda = 1$)	7M	7.44	32.74	–
ResNet50 ($\lambda = 0$)	25.6M	–	–	30.98
ResNet50-Cache1	25.6M	–	–	**30.42**
ResNet50-Cache1-CacheOnly ($\lambda = 1$)	25.6M	–	–	41.24

Adversial attack

	FGSM	I-FGSM	SP	GB
$\langle \rho_{\text{adv}} \rangle$	0.064	0.014	0.044	0.224
ResNet32 ($\lambda = 0$)	5.2	5.2	9.8	2.8
ResNet32-Cache3	**72.8**	68.4	58.8	56.0
ResNet32-Cache3-CacheOnly ($\lambda = 1$)	72.0	**83.2**	**68.6**	**61.3**

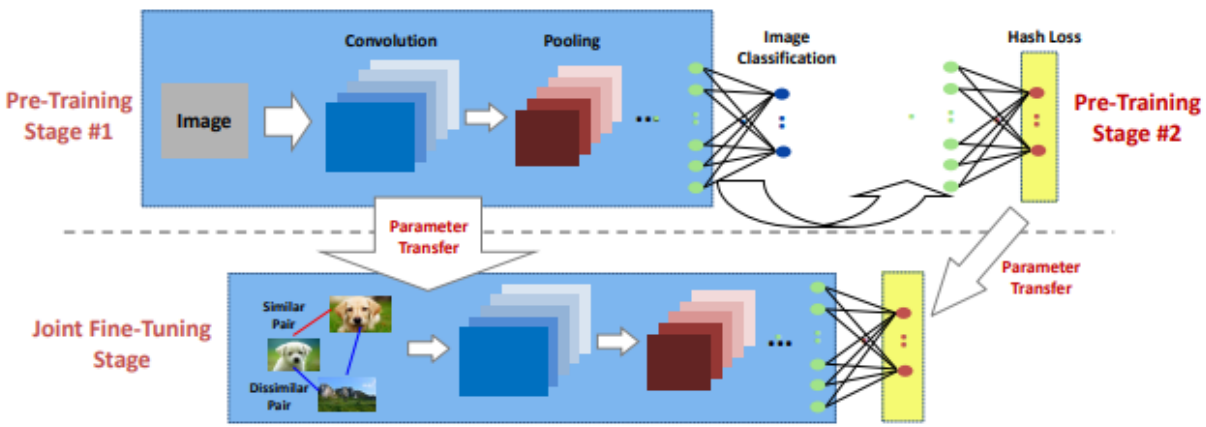
Key technical insights

Key observations:

- Layers of a network close to the output layer could contain easily extractable class-relevant information that is not already contained in the output layer itself.
- In the paper, it is shown that the cache-only model performs poorly than the baseline model.
- The cache-only model tends to be more robust to input pertubations compared to the baseline model and the linear-combination cache model.
- Significant improvements in test accuracy over the baseline model can be observed even with small cache sizes.
- All these observations are limited to networks with skip connections.

Key technical insights

Hashing technique : Deep Hashing, A Joint Approach for Image Signature Learning



A DL based framework is taken as the main network and piece of the network is used for hashing. Vectors are obtained across various layers of the hash network for cache comparison. The hash function is then trained for the second time to ensure that similar input images have similar hash functions and dissimilar images do not.

Key technical insights

- The final hash layer is sent through a signum function to get an output from the set {-1, 0, 1}.
- The output hash codes are using exponential loss function(it was suggested that triplet loss is very complex to implement) inorder to reduce the hamming distance between hash values for similar inputs.

$$\ell(\mathbf{x}_i, \mathbf{x}_j) = e^{-\mathbf{Y}_{i,j}(\mathbf{b}_i \circ \mathbf{b}_j)}.$$

where $Y_{.,.}$ denotes the signum function

- The gradient computation for the hash layers are approximated extrinsically for back-propagation due to the presence of signum function.

Key technical insights

Results:

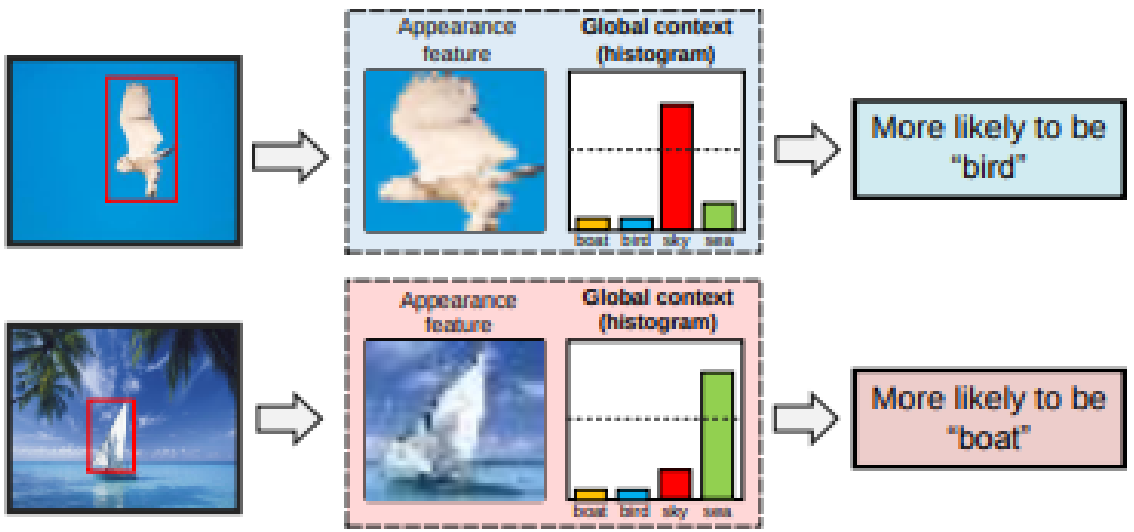
	MNIST			CIFAR10			Kaggle-Face			SUN397		
	12 bits	24 bits	48 bits	12 bits	24 bits	48 bits	12 bits	24 bits	48 bits	12 bits	24 bits	48 bits
DeepHash (random init.)	0.9806	0.9862	0.9873	0.5728	0.6503	0.6585	0.4125	0.4473	0.4620	0.0211	0.0384	0.0360
DeepHash (pre-training)	0.9673	0.9753	0.9796	0.4986	0.5588	0.5966	0.4282	0.4484	0.4589	0.0335	0.0430	0.0592
DeepHash (fine-tuning)	0.9918	0.9931	0.9938	0.6874	0.7289	0.7410	0.5487	0.5552	0.5615	0.0748	0.1054	0.1293

Key observation:

- The two stage of training(catg. 3 in table) tends to perform better than other learning techniques (catg. 1 and 2) for hashing networks.
- The fine-tuning based learning procedure also outperforms other state-of-the-art techniques performed in hashing.

Key technical insights

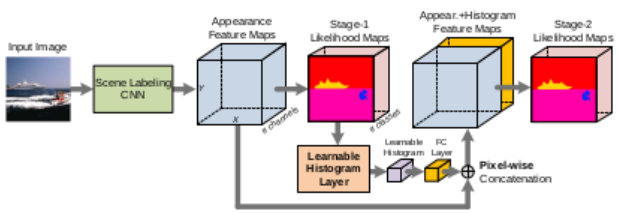
Hash function : Learnable Histogram, Statistical Context Features for Deep Neural Networks



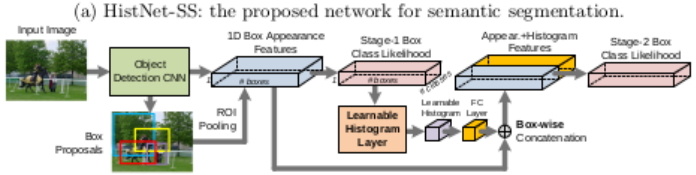
- An histogram with fixed number of bins for each class is learned during the training process and this learned histogram will be used for classification.

Key technical insights

The general architecture for the HistNet consists of a baseline model over which histogram layer is applied followed by fully connected layer.

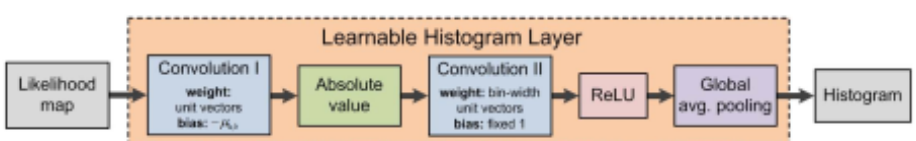


(a) HistNet-SS: the proposed network for semantic segmentation.



(b) HistNet-OD: the proposed network for object detection.

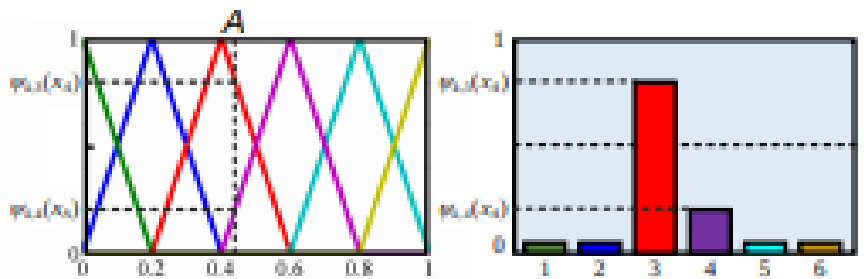
An histogram layer for efficient back-propagation can be approximated into convolutional layers with bin widths and centers learnt as the weights of the network.



Key technical insights

- The probability for each bin of a specific class is given by the following formula:

$$\psi_{k,b}(x_k) = \max\left\{0, 1 - \frac{1}{w_{k,b}} \times |x_k - \mu_{k,b}|\right\}$$



- The parameters $\mu_{k,b}$ and $w_{k,b}$ denote the bin center and bin width respectively. These parameters are learnt during training of the network. The gradients used for back propagation on learning is:

$$\frac{\partial E}{\partial u_{k,b}} = \begin{cases} w_{k,b}, & \psi_{k,b}(x_k) > 0 \text{ and } x_k - \mu_{k,b} > 0, \\ -w_{k,b}, & \psi_{k,b}(x_k) > 0 \text{ and } x_k - \mu_{k,b} < 0, \\ 0, & \text{otherwise.} \end{cases}$$

$$\frac{\partial E}{\partial w_{k,b}} = \begin{cases} |x_k - \mu_{k,b}|, & \psi_{k,b}(x_k) > 0, \\ 0, & \text{otherwise.} \end{cases}$$

Key technical insights

Results:

Methods	Per-pixel	Per-class
Tighe et al. [32]	0.769	0.294
Liu et al. [25]	0.748	n/a
Farabet et al. [12]	0.785	0.296
Pinheiho et al. [18]	0.777	0.298
Sharma et al. [29]	0.796	0.336
Yang et al. [1]	0.798	0.487
Eigen et al. [31]	0.868	0.464
FCN [19]	0.851	**0.517**
FCN (our implement)	0.860	0.457
FCN+FC-CRF	0.865	0.468
HistNet-SS stage-1	0.876	0.505
HistNet-SS	**0.879**	0.5
HistNet-SS+FC-CRF	**0.879**	0.512

(a) SIFTFlow dataset

Method	Per-pixel	Per-class
Gould et al. [2]	0.764	n/a
Tighe et al. [32]	0.775	n/a
Socher et al. [28]	0.781	n/a
Lempitzky et al. [33]	0.819	0.724
Farabet et al. [12]	0.814	0.76
Pinheiho et al. [18]	0.802	0.699
Sharma et al. [29]	0.823	0.791
FCN (our implement)	0.851	0.811
FCN+FC-CRF	0.862	0.82
FCN+MOPCNN [17]	0.863	0.811
HistNet-SS stage-1	0.871	**0.838**
HistNet-SS	0.871	0.837
HistNet-SS+FC-CRF	**0.881**	0.837

(b) Stanford background dataset

Methods	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow
RCNN [11]	73.4	77.0	63.4	45.4	44.6	75.1	78.1	79.8	40.5	73.7
fast RCNN [36]	74.5	78.3	69.2	53.2	36.6	77.3	78.2	82.0	40.7	72.7
faster RCNN [23]	69.1	78.3	68.9	55.7	49.8	77.6	79.7	85.0	51.0	76.1
HistNet-OD stage-1	68	80.3	74.1	55.7	53.3	83.6	80.2	85.1	53.7	74.2
HistNet-OD	67.6	80.3	74.1	55.6	53.2	83.4	80.2	85.1	53.6	74

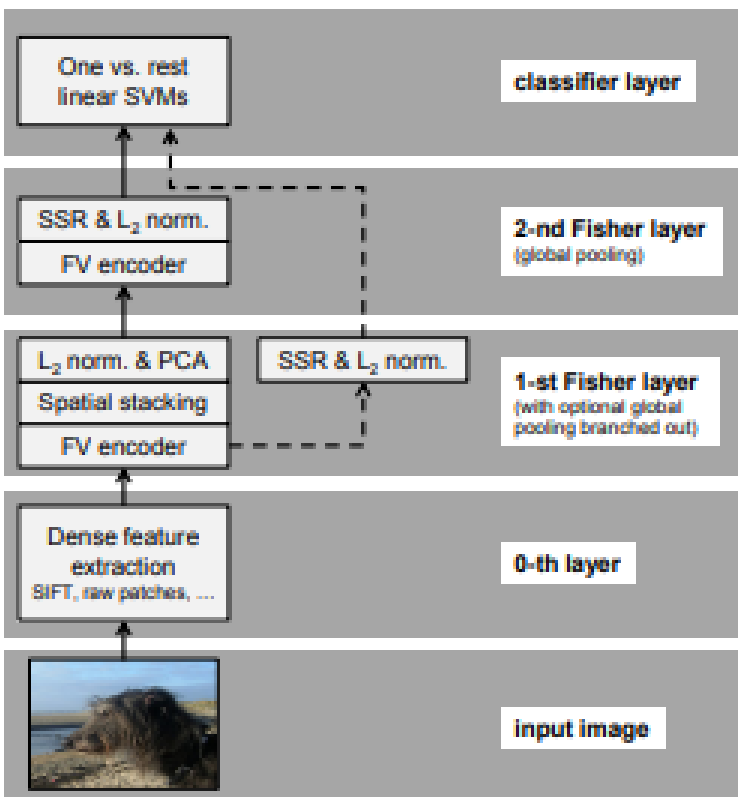
	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	**mAP**
RCNN [11]	62.2	79.4	78.1	73.1	64.2	35.6	66.8	67.2	70.4	71.1	66.0
fast RCNN [36]	67.9	79.6	79.2	73.0	69.0	30.1	65.4	70.2	75.8	65.8	66.9
faster RCNN [23]	64.2	82.0	80.5	76.2	75.8	38.5	71.4	65.4	77.8	66.1	69.5
HistNet-OD stage-1	69.3	82.5	84.9	76.5	77.7	44.2	71.7	66.6	75.5	71.8	71.4
HistNet-OD	69.3	82.5	84.8	76.3	77.6	44.1	71.9	66.8	75.4	71.9	71.4

Observations:

- Two phase incremental training of Histogram networks tend to out-perform the state-of-the-art models in object detection and semantic segmentation tasks.
- By training with the learnable histogram layer, the base network is also improved by jointly finetuning. After fine-tuning with histograms, the original base model can also be improved, which suggest a new way for training.

Key technical insights

Hash function : Deep Fisher Networks for Large-Scale Image Classification



The network uses fisher vector layer, which is a generalization of the standard FV to a layer architecture suitable for stacking.

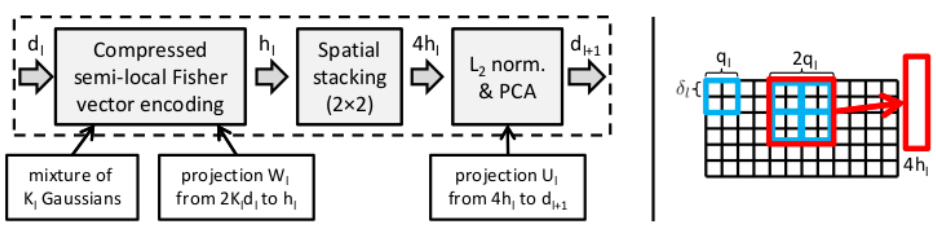
By discriminatively training several such layers and stacking them into a Fisher Vector Network, an accuracy competitive with the deep CNN can be achieved, whilst staying in the realms of conventional SIFT and colour features and FV encodings

Key technical insights

The Fisher vector encoding ϕ of a set of features,x_p(densely computed SIFT features) is based on fitting a parametric generative model, like Gaussian Mixture Model (GMM), to the features, and then encoding the derivatives of the log-likelihood of the model with respect to its parameters.

$$\Phi_k^{(1)} = \frac{1}{N\sqrt{\pi_k}} \sum_{p=1}^{N} \alpha_k(x_p) \left(\frac{x_p - \mu_k}{\sigma_k} \right), \quad \Phi_k^{(2)} = \frac{1}{N\sqrt{2\pi_k}} \sum_{p=1}^{N} \alpha_k(x_p) \left(\frac{(x_p - \mu_k)^2}{\sigma_k^2} - 1 \right)$$

Spatial stacking is performed to extract the spatial information of the image. Capturing such spatial information makes the prediction more accurate and also makes the representation invariant to small translations.



Key technical insights

Results:

pipeline setting	SIFT only			SIFT & colour	
	dimension	top-1	top-5	top-1	top-5
1st Fisher layer	82K	46.52	68.45	55.35	76.35
2nd Fisher layer	131K	48.54	71.35	56.20	77.68
multi-layer (1st and 2nd Fisher layers)	213K	**52.57**	**73.68**	**59.47**	**79.20**
Sánchez et al. [23]	524K	N/A	67.9	54.3	74.3

Key observations:

- The network reaches close to performance of primitive convolutional network(like AlexNet) with very fewer number of stacked layers and parameters.
- The convolutional networks and Fisher vector encodings are complementary in the sense that their combination further improves the accuracy.

Key technical insights

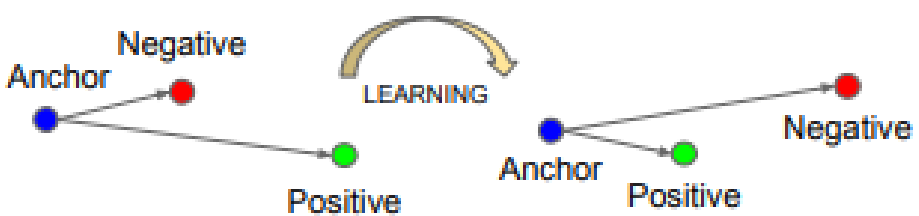
Triplet loss:

The Triplet Loss minimizes the distance between anchor and positive examples, both of which have the same class, and maximizes the distance between the anchor and negative examples, which is of a different class.

Given an anchor , a set of positive examples and negative examples,

$$Loss = max(||anchor - positive||) - min(||anchor - negative||) + margin$$

where ||.|| denotes the Euclidean norm of the vector



Key technical insights

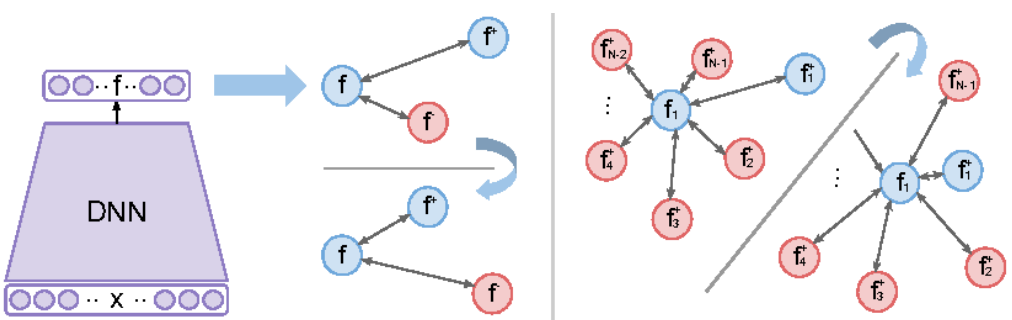
N-Pair loss:

Unlike triplet loss, N-Pair loss tries to push a set of negative examples away from the anchor rather than just one.

Given an anchor , a set of positive examples and negative examples,

$$Loss = margin - \log_e \left(1 + \sum_{i=1}^{N-1} \exp\left(||negative[i] - anchor|| - ||positive - anchor||\right)\right)$$

where ||.|| denotes the Euclidean norm of the vector



Key technical insights(modifications)

Application of signum function can lead to generation of cache keys with poor information about the input image. Hence, signum function is not used. This allows us to use triplet and n-pair losses.

For networks different from the baseline model, the two stages of training carried out are:

- Coarse training stage : The hash network is trained to ensure that the cachable vector obtained from the hash network is similar to the cachable vector obtained from the cache network.
- Fine-tuning stage : The network is trained to ensured that similar input images when provided as input to the network produces vector representations that are similar to each other.

Cache is not updated while training different hash functions.

Key technical insights(modifications)

In case of the HistNet, the networks histogram is given as an input to fully connected layers. These fully connected layers are trained during the coarse training stage and fine-tuning stage.

For FisherNet, the Gaussian Mixture Models are stacked as the weights of a convolutional network and the parameters are learnt during the training stage. Also, dimensionality reduction is not performed to avoid information loss. On the other hand, spatial stacking also is not performed, because spatial stacking leads to a significant increase in the inference time and also makes the training process extremely slow.

For CNN and HistNet model, the pretrained model is taken and the fully connected layers used to generate cache are only trained. But for FisherNet, this technique didn't work. Hence, the whole model is trained for FisherNet.

By Mid-term

```
def train_cache_knn(trainloader):
  net_hash.train()
  train_loss = 0
  total = 0
  for batch_idx, (inputs, targets) in enumerate(trainloader):
    lists = []
    inputs, targets = inputs.to(device), targets.to(device)
    optimizer_hash.zero_grad()
    vector, _ = net_hash(inputs)
    vector = vector.squeeze()
    vector1 = net_cache(inputs)
    loss = criterion_hash(vector, vector1)
    loss.backward()
    optimizer_hash.step()
    train_loss += loss.item()
    total += targets.shape[0]
```

Coarse training stage

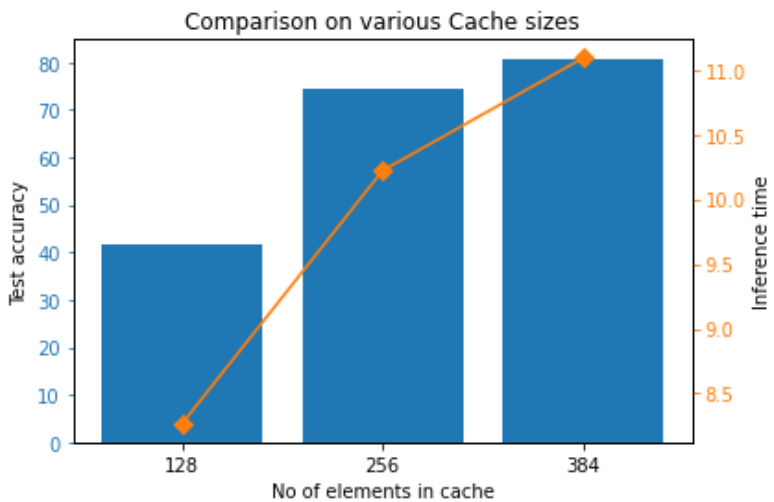
By Mid-term

```
def train_cache_knn(trainloader):
  net_hash.train()
  train_loss = 0
  correct = 0
  total = 0
  for batch_idx, (inputs, targets) in enumerate(trainloader):
    lists = []
    inputs, targets = inputs.to(device), targets.to(device)
    optimizer_hash.zero_grad()
    vector, _ = net_hash(inputs)
    diff = cache_val - vector
    norm = torch.norm(diff, p=1, dim=1)
    loss = criterion_hash(vector, targets, norm)
    loss.backward()
    optimizer_hash.step()
    train_loss += loss.item()
    ind = (torch.topk(norm, 50, largest=False)[1])
    for index in ind:
      lists.append(cache_key[index])
    freq, predicted = mode(lists)
    total += targets.shape[0]
    correct += (predicted == int(targets))
```

Fine training stage

By Mid-term

The accuracy and execution time of the network for classification on CIFAR10 dataset when cache is taken at the end of various baics blocks of a ResNet-18 model:

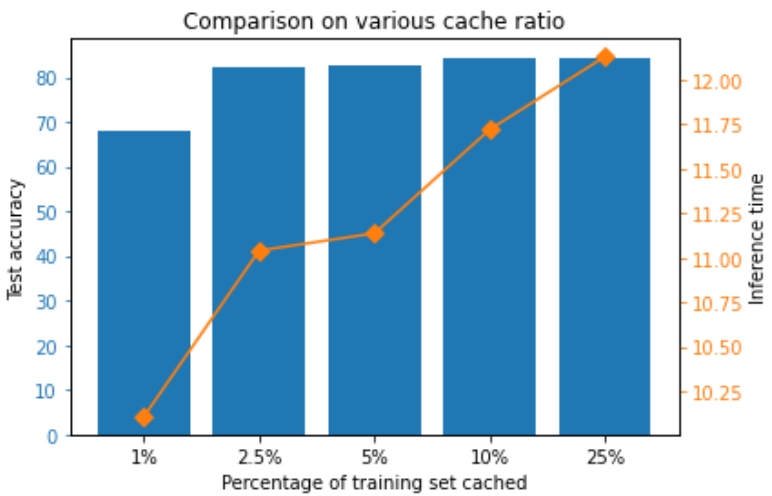


The cache with 128 and 256 elements are cached at the end of 2nd and 3rd basic blocks respectively. The 384 elements cache is taken from both blocks.

Caching at very earlier stage leads to a significant drop in accuracy. Considering accuracy-time trade-off 256 elements cache is selected for analysing the effect of cache ratio.

After Mid-term

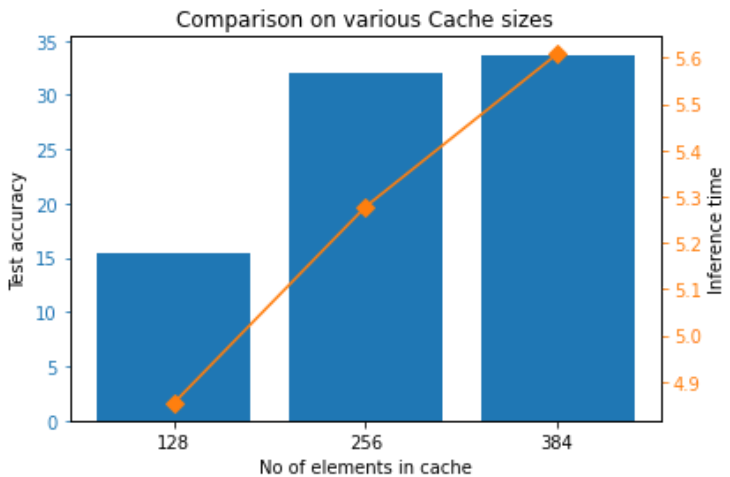
The accuracy and execution time of the network for classification on CIFAR10 dataset when cache is taken at the end of 2nd basic block for various cache ratio in training set:



On increasing cache ratio above 2.5%, the improvement in accuracy is very less compared to the increase in inference time. Hence the cache ratio is taken as 2.5% for training hash functions.

After Mid-term

The accuracy and execution time of the network for classification on CIFAR10 dataset when cache is taken at the end of various baics blocks of a VGG-16 model:

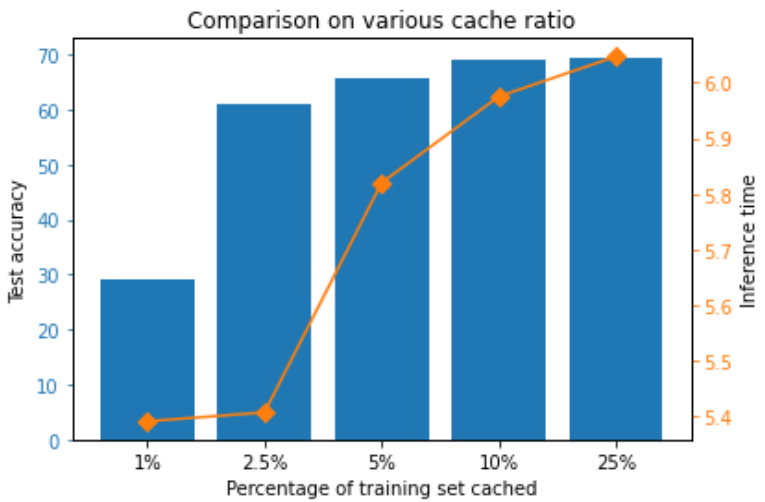


The cache with 128 and 256 elements are cached at the end of 2nd and 3rd basic blocks respectively. The 384 elements cache is taken from both blocks.

Caching at very earlier stage leads to a significant drop in accuracy. Considering accuracy-time trade-off 256 elements cache is selected for analysing the effect of cache ratio.

After Mid-term

The accuracy and execution time of the network for classification on CIFAR10 dataset when cache is taken at the end of 2nd basic block for various cache ratio in training set:



On increasing cache ratio above 5%, the improvement in accuracy is very less compared to the increase in inference time. But test accuracy for all combination of hyper-parameters is lower than the test accuracy observed by caching on ResNet-18.

After Mid-term

Different hash functions used in this project are:

- Broken ResNet-18(part of ResNet-18)
- Custom defined 6-layer CNN
- Learnable Histogram based classification
- Deep Fisher network

Different loss functions used in this project are:

- Triplet loss
- N-Pair loss

Tests carried out on hash functions are:

- Adversial attacks(FGSM attack)
- Variation in brightness
- Variation in contrast
- Variation in gamma-correlation

After Mid-term

The 4 models described above are defined here:

- Broken ResNet-18:

  This network contains the first three basic blocks of the original ResNet-18 model. Since it contains only part of the total model, the inference time is expected to reduce while maintaining test accuracy if trained efficiently.

- Custom 6-layer CNN:

  This network consists of six convolutional layers followed by fully connected layers. The output of this network is a vector with dimension same as the cache values. The network when trained efficiently is expected to mimic the function mapped by ResNet-18 with much fewer parameters and computation, reducing inference time.

After Mid-term

- HistNet:

  This network consists of six convolutional layers followed by one histogram layers and two fully connected layers. The output of this network is a vector with dimension same as the cache values. The network when trained efficiently is expected to mimic the function mapped by ResNet-18 with much fewer parameters and computation, reducing inference time.

- FisherNet:

  This network consists of four fisher layers stacked above one another followedd by two fully connected layers. The input to this model is converted into SIFT features and then and passed on to the Fisher layers. The output of this network is a vector with dimension same as the cache values.

After Mid-term

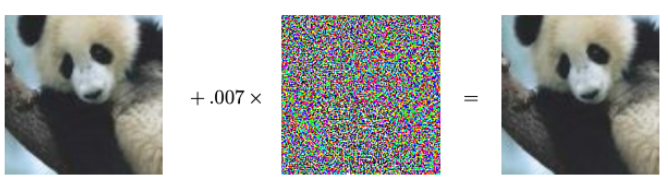
The link to onnx structure of the models are provided below:

https://drive.google.com/drive/folders/1VQpjoDxhs12ALgQDhIje9J5TmdpTiLpo?usp=sharing

All the models are trained for 5 epochs in both the stages of training.

Adversial attack(FGSM attack):

Fast Gradient Sign Method attack is a white box attack whose goal is to ensure misclassification. A white box attack is where the attacker has complete access to the model being attacked.The method uses the gradients of the loss with respect to the input image to create a new image that maximises the loss. This new image is called the adversarial image.

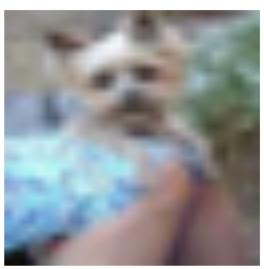


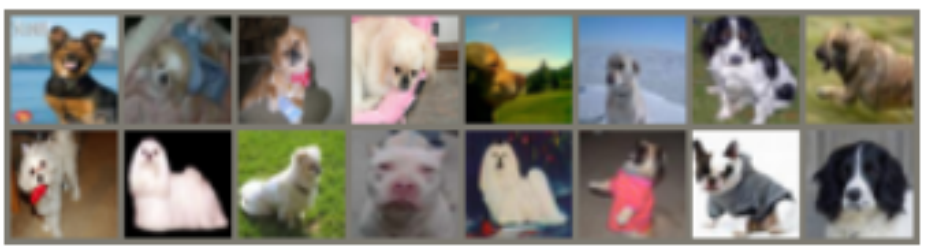
After Mid-term

Broken ResNet-18 with Triplet loss:

One obvious hash function that can reduce the network inference time is a broken ResNet-18 network with parameters transferred from the original network and fine tuned on training with Triplet loss.

Cat





Frog

Inputs

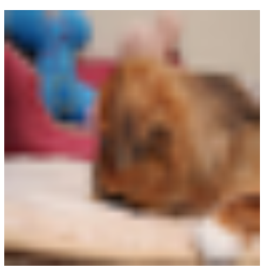
Nearest neighbours

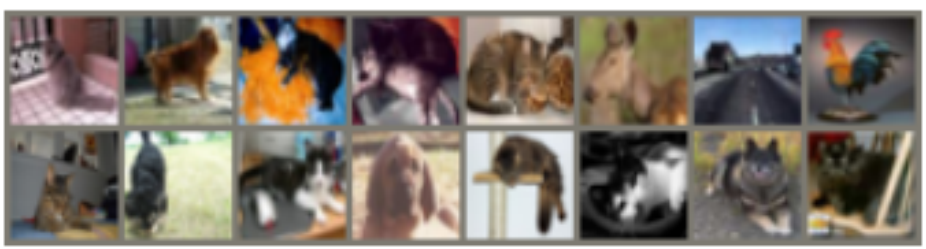
After Mid-term

Broken ResNet-18 with N-Pair loss:

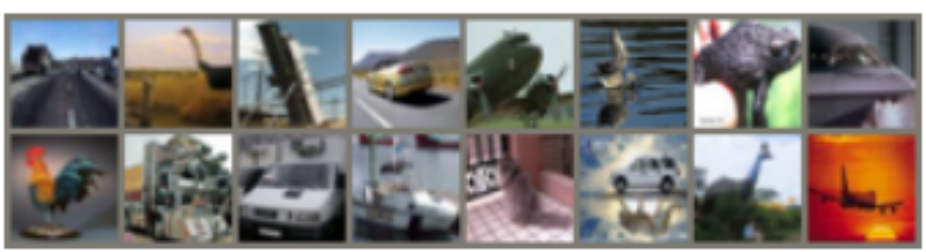
Now the Broken ResNet-18 network is used with the same initialization as described before and trained with N-Pair loss during the fine tuning stage.

Cat





Truck



Inputs

Nearest neighbours

Tabulation of Results:

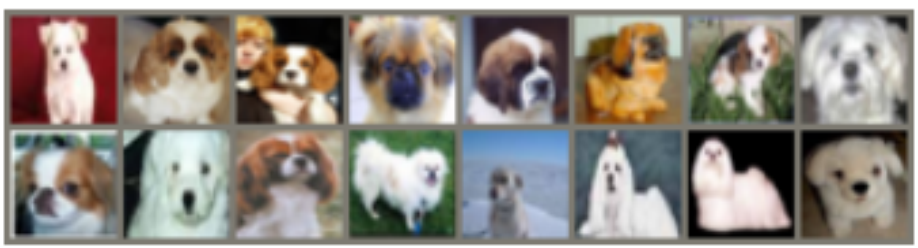
Network ->	**ResNet-18**	**B-ResNet(TL)**	**B-ResNet(NL)**
Test Accuracy	87.3%	**88.0%**	86.2%
FGSM(ε=0.05)	44.2%	**75.2%**	68.3%
FGSM(ε=0.1)	24.7%	**68.2%**	59.2%
Contrast(c=0.5)	77.6%	**78.3%**	73.1%
Contrast(c=1.5)	80.2%	**83.1%**	77.1%
Brightness(b=0.5)	78.0%	**79.1%**	67.0%
Brightness(b=1.5)	78.1%	**80.6%**	72.3%
γ-corrected(γ=0.5)	74.9%	**78.3%**	71.6%
γ-corrected(γ=1.5)	80.6%	**83.9%**	69.7%
Inference time(1000 images)	21.26 secs	12.84 secs	**12.82 secs**

After Mid-term

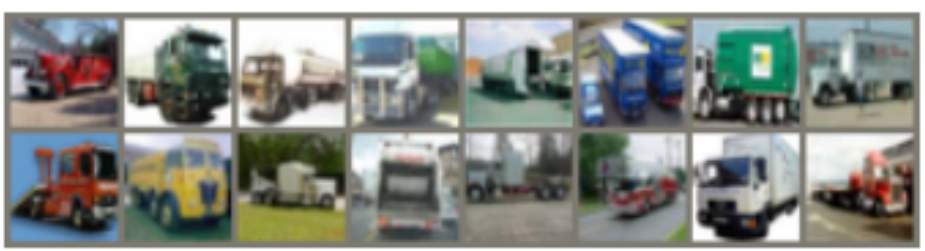
6-layer CNN with Triplet loss:

The same 6-layer CNN is taken with the same initialization technique followed before but is instead trained with Triplet loss in the Fine tuning stage.

Dog



Truck



Inputs

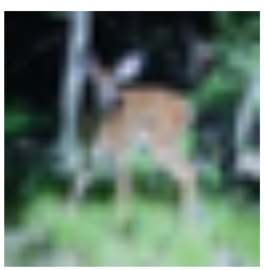
Nearest neighbours

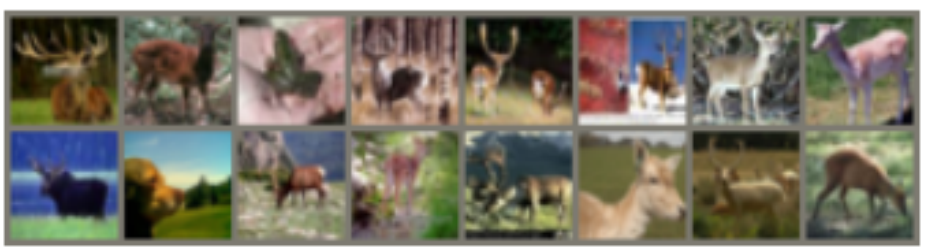
After Mid-term

6-layer CNN with N-Pair loss:

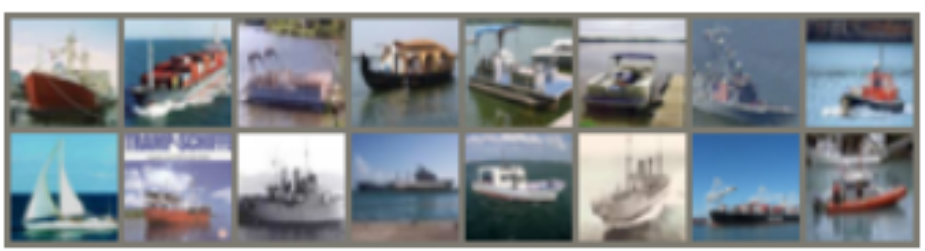
An other possible hash function is a simple Convolutional Network which can be used to mimic the ResNet-18 network. The initial parameters of the network are once transferred from pretrained 6-layer CNN with two FCs. The network is trained with N-Pair loss.

Deer





Ship



Inputs

Nearest neighbours

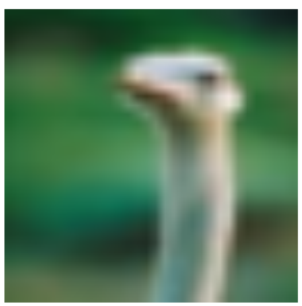
Tabulation of Results:

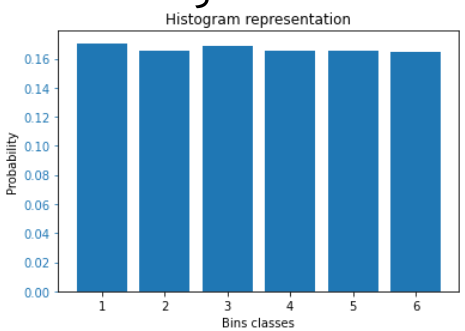
Network ->	**ResNet-18**	**ConvNet(TL)**	**ConvNet(NL)**
Test Accuracy	**87.3%**	86.4%	79.5%
FGSM(ε=0.05)	44.2%	**75.4%**	70.3%
FGSM(ε=0.1)	24.7%	**68.3%**	63.8%
Contrast(c=0.5)	**77.6%**	73.0%	70.2%
Contrast(c=1.5)	**80.2%**	77.9%	75.2%
Brightness(b=0.5)	78.0%	**78.3%**	69.9%
Brightness(b=1.5)	78.1%	**79.4%**	74.3%
γ-corrected(γ=0.5)	74.9%	**78.3%**	74.5%
γ-corrected(γ=1.5)	**80.6%**	79.0%	75.1%
Inference time(1000 images)	21.26 secs	**13.12 secs**	13.29 secs

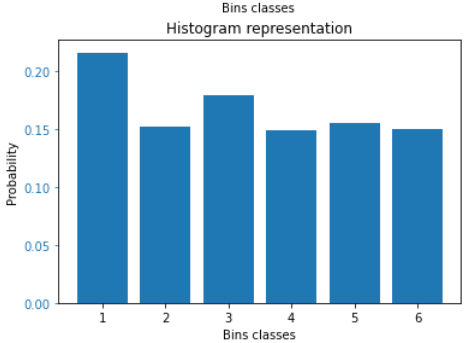
After Mid-term

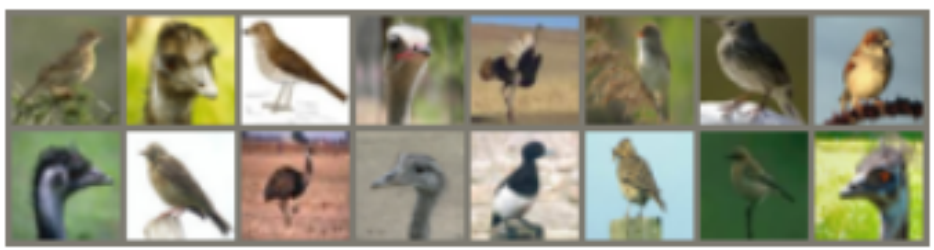
HistNet with Triplet loss:

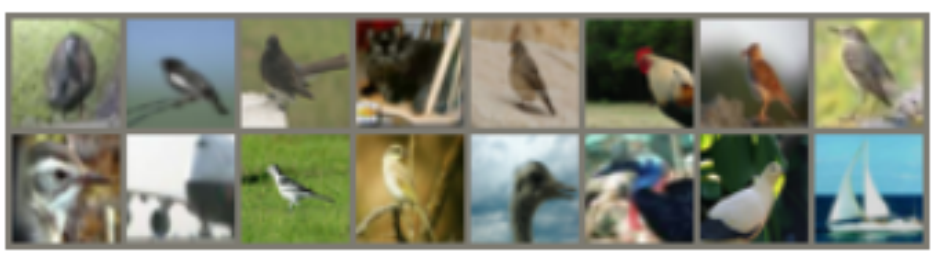
An histogram network combined with simple convolutional layers can be used to extract intra-class features and compare images within the class. A different histogram is built for each output class with 6 bins.









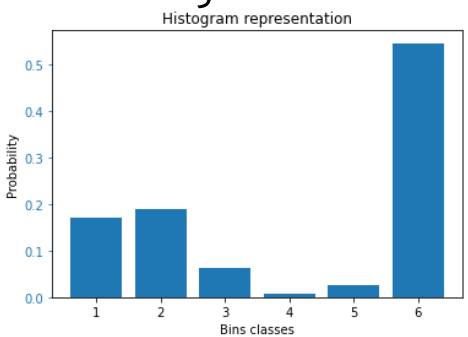


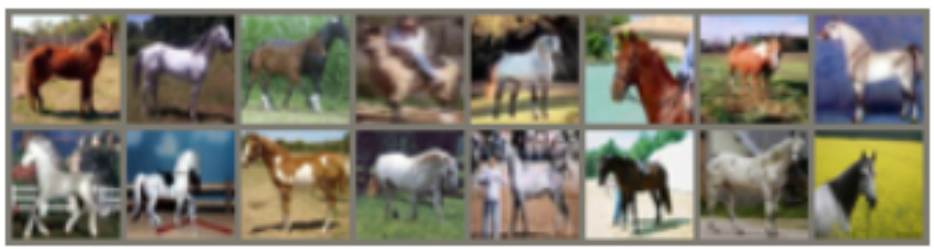
Inputs Histogram Nearest neighbours

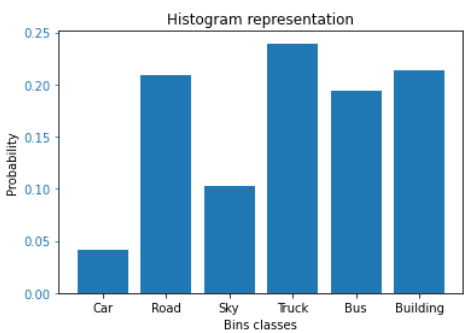
After Mid-term

HistNet with N-Pair loss:

An histogram network combined with simple convolutional layers can be used to extract intra-class features and compare images within the class. A different histogram is built for each output class with 6 bins.







Inputs Histogram Nearest neighbours

Tabulation of Results:

Network ->	**ResNet-18**	**HistNet(TL)**	**HistNet(NL)**
Test Accuracy	87.3%	**88.6%**	**88.6%**
FGSM(ε=0.05)	44.2%	74.8%	**75.2%**
FGSM(ε=0.1)	24.7%	**69.2%**	69.1%
Contrast(c=0.5)	**77.6%**	76.4%	77.1%
Contrast(c=1.5)	**80.2%**	79.8%	76.5%
Brightness(b=0.5)	**78.0%**	67.3%	68.8%
Brightness(b=1.5)	78.1%	**80.6%**	78.3%
γ-corrected(γ=0.5)	74.9%	**80.4%**	78.8%
γ-corrected(γ=1.5)	**80.6%**	78.7%	78.0%
Inference time(1000 images)	21.26 secs	**9.28 secs**	10.36 secs

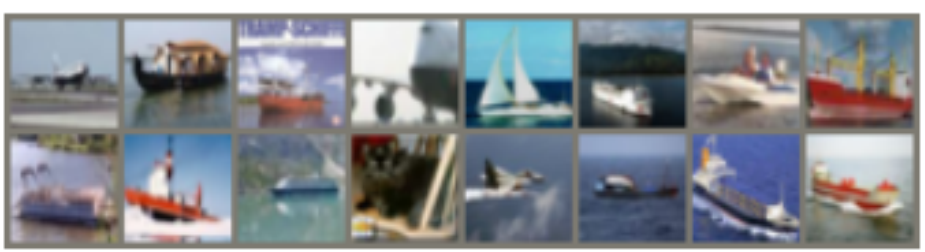
After Mid-term

FisherNet with Triplet loss:

One other hash function used in this project is the Deep Fisher Network. In this network, the SIFT features are given as inputs and fisher network is trained for the classification using cache with triplet loss function.

Cat

Frog



Inputs

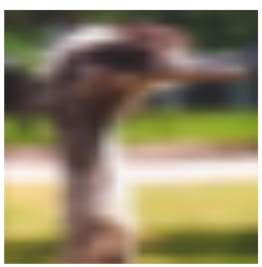
Nearest neighbours

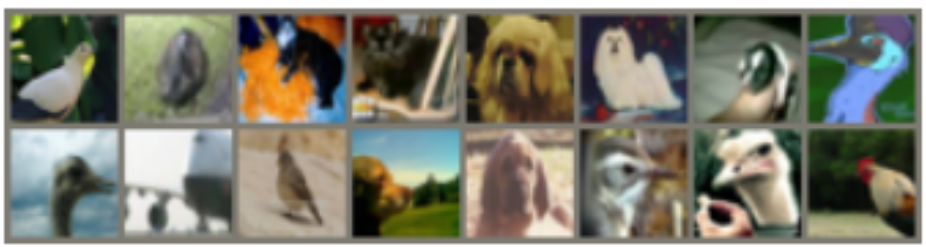
After Mid-term

FisherNet with N-Pair loss:

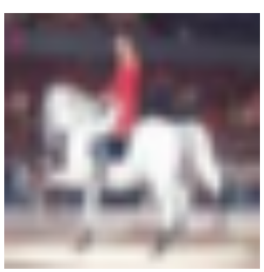
One other hash function used in this project is the Deep Fisher Network. In this network, the SIFT features are given as inputs and fisher network is trained for the classification using cache with N-Pair loss function.

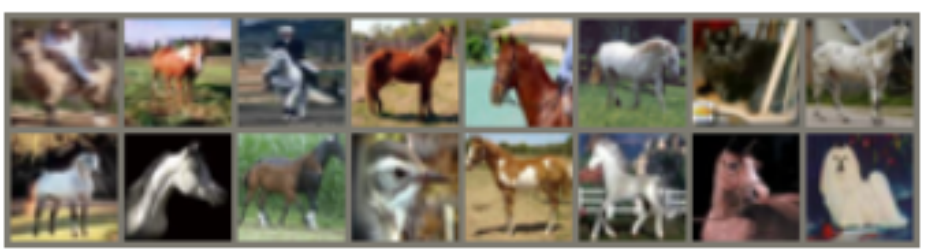
Bird





Horse





Inputs

Nearest neighbours

Tabulation of Results:

Network ->	**ResNet-18**	**FisherNet(TL)**	**FisherNet(NL)**
Test Accuracy	**87.3%**	46.9%	48.2%
FGSM(ε=0.05)	**44.2%**	33.5%	35.6%
FGSM(ε=0.1)	24.7%	31.3%	**35.2%**
Contrast(c=0.5)	**77.6%**	30.3%	30.6%
Contrast(c=1.5)	**80.2%**	36.7%	38.8%
Brightness(b=0.5)	**78.0%**	30.3%	31.0%
Brightness(b=1.5)	**78.1%**	37.2%	37.0%
γ-corrected(γ=0.5)	**74.9%**	21.5%	17.0%
γ-corrected(γ=1.5)	**80.6%**	78.7%	31.8%
Inference time(1000 images)	21.26 secs	**14.32 secs**	14.83 secs

Observations(specific to ref. papers):

- As suggested in the Simple Cache paper, the cache models tend to make the predictions more robust and also increases the generality in the test set.
- In contrary, when hash functions for these cache only models are trained, they tend to give very close or better performance when compared to the baseline model in the test set.
- As suggested in the Deep Hashing paper, the two level training with fine-tuning leads to highly accurate hash models from the same baseline models.
- This type of training is extended to train different hash functions with a coarse training stage preceding the fine-tuning stage to reduce the L2/L1 norm between the cache and hash models.

Observations(specific to ref. papers):

- The added histogram layers in the HistNet improves the performance of the custom defined 6-layer conv network. The histogram network also outperforms the baseline model and also the hash model derived from baseline model.
- As seen in the FisherNet paper's results, the models accuracy for image classification is pretty low compared to other Deep convolutional networks. But unlike convolutional networks, the number of parameters in the FisherNet is very low and when optimized properly could lead to much faster inference compared to deep conv net.

Observations:

- Cache obtained from ResNet-18 shows better accuracy than cache values obtained from VGG-16. This could be due to the skip connections in ResNet-18 which leads to better information transfer between layers compared to simple feed-forward network incorporated in VGG-16.
- Caching in general with appropriate hash functions lead to test accuracy comparable or better than the original network.
- A simple 6-layer network being able to mimic the ResNet-18 suggests that even small networks when trained appropiately, using loss functions like Triplet loss and N-Pair loss instead of Cross Entropy loss might give much better accuracy for Image classification.

Observations:

- The Histogram network's bin generation for every class uniquely helps in indenifying intra-class relations, like both jets and aeroplane fall in the airplanes class in CIFAR-10 which can be categorized further with this network.
- Caching reduces the effect of adversial attacks, change in environmental conditions like contrast, brightness and γ-correction.
- The histogram hash function gives very high test accuracy with minimal inference time when trained with Triplet loss making it the best hash function.
- The fisher network fails to provide high accuracy hash models.
- Triplet loss function in general produces more accurate hash functions than N-Pair loss function. The ability of Triplet loss function to bring similar inputs cose to each other is better than that of N-Pair loss.

Future Works(specific to ref. papers):

- The cache used for the project is limited to the output of the third resnet block. More experimentations like using the output of the final block and also dimensionality reduction could lead to better performance and also faster inference.
- The histogram layers can be trained with hard labels instead of adding intermediate fully connected layers which could lead to better performance in intra-class classification.
- The accuracy of the fisher network can be improved by performing efficient spatial stacking making deep net.

Note : Since Fisher Networks and Convolutional networks are observed to be complementary, it is not possible to get models with accurates close to the HistNet based models and others.

Future works:

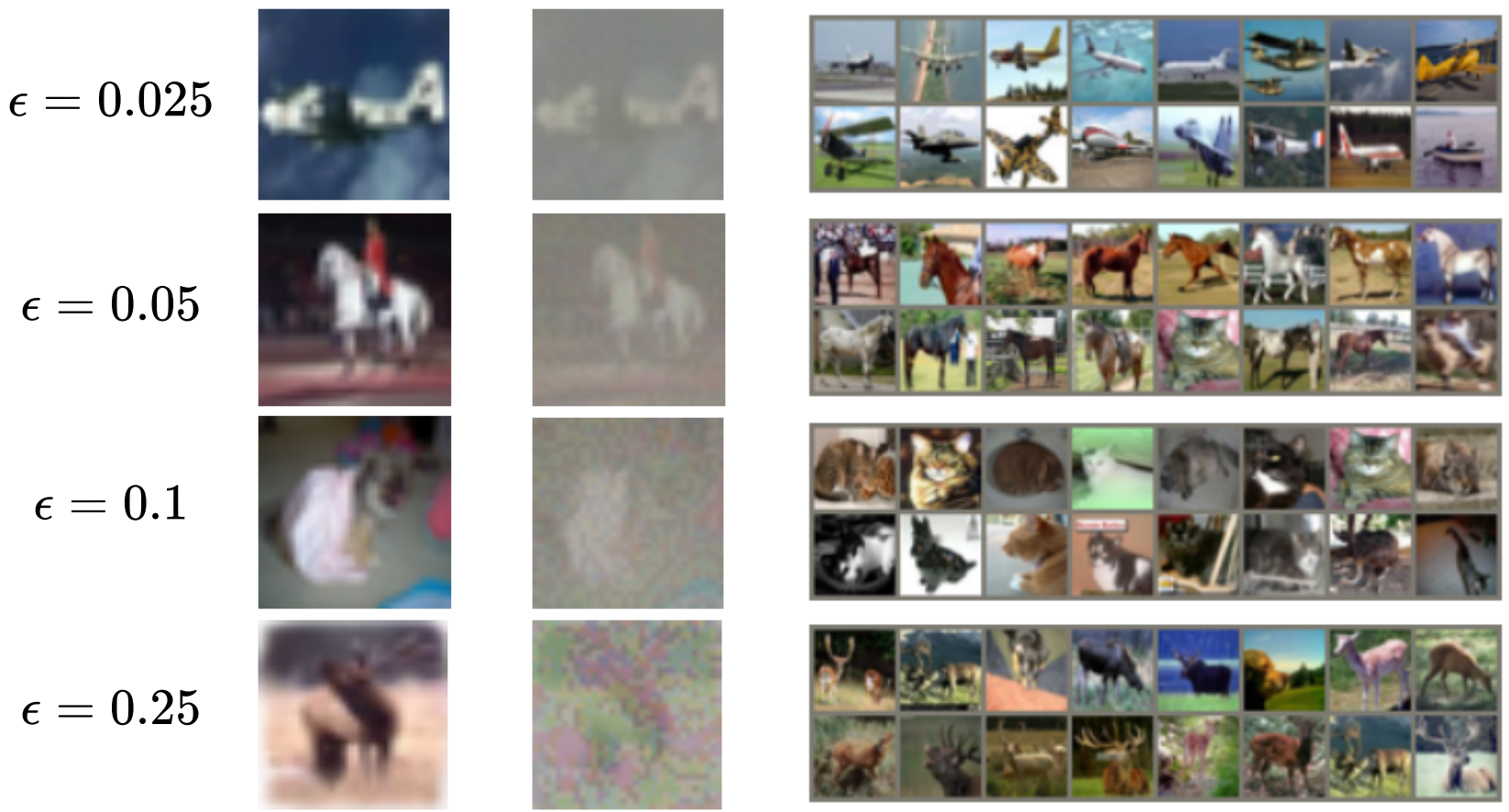
- With just 2.5% of training set used for caching, the memory taken by the cache file is approximately 1.3MB and with hash functions like HistNet and FisherNet which use only one-fourth the number of parameters used by ResNet-18, these networks can be easily stored in edge devices compared to deep neural networks.
- The ability of Histogram and 6-layer Convolution network to give high accuracy with reduced inference time shows that these networks can be processed in edge and mobile devices for much faster inference in real time.
- This would mean that an implementation of these networks in edge devices could be a task for the future.
- Also such networks can be extended for tasks such as object detection, semantic segmentation and so on.

Future works:

- Testing the networks with larger datasets, like ImageNet dataset to check the model performance.
- Also it can be noticed that hash models with appropiate cache keys and values are unsusceptible to lighting condition, contrast change and also to moderate level of adversial attacks.
- This implies that such models can be used for AI applications like in case of self-driving cars for which variation in climatic conditions and other environmental variable prove to be a major obstacle in efficient classification and detection tasks.
- Testing the ability of caching and the use of hash functions to various other computer vision like semantic segmentation and objection detection for improve accuracy and reduce inference time.

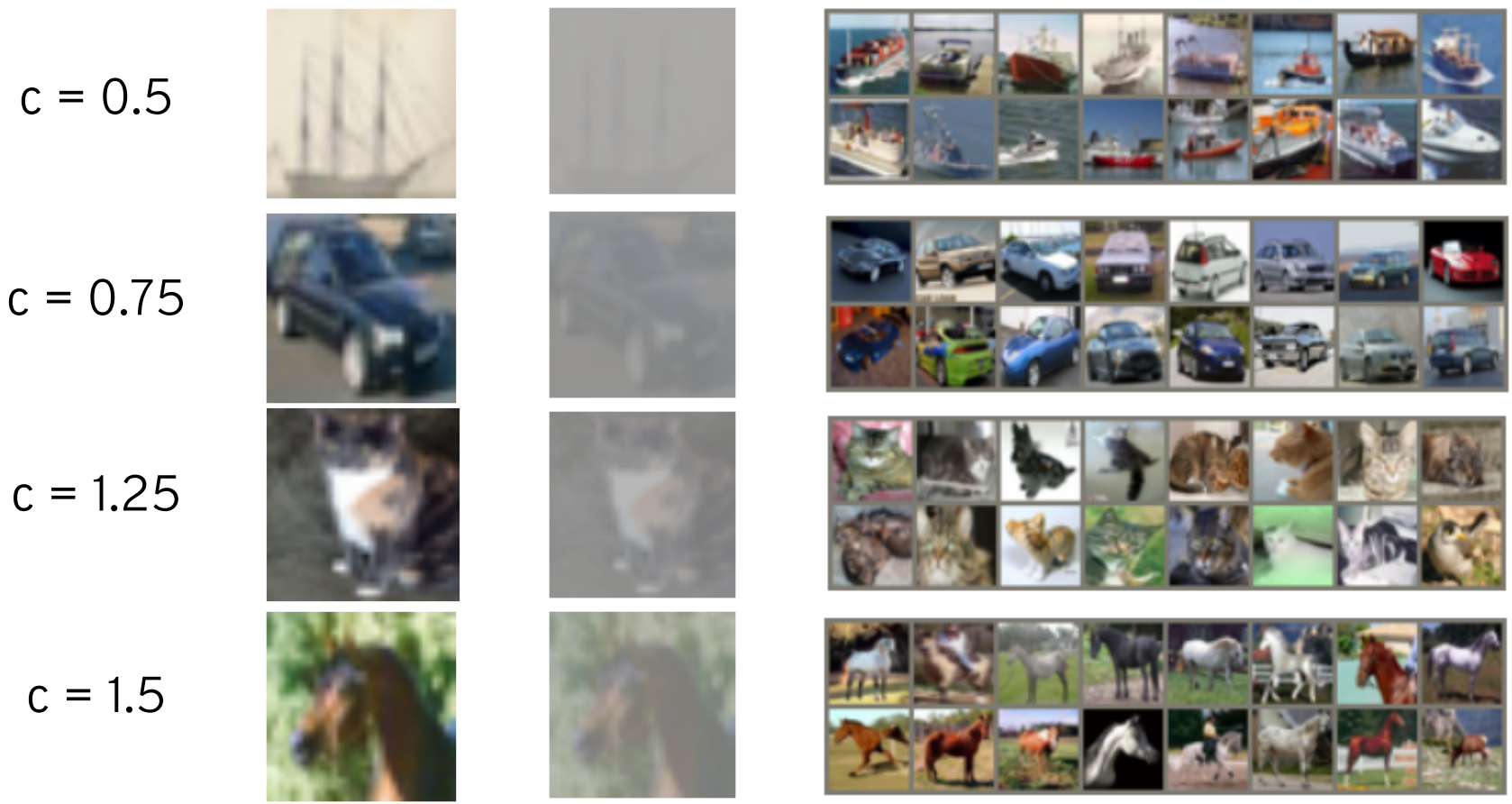
Appendix

Broken ResNet-18 with Triplet Loss: Effects of Adversial attack(FGSM)



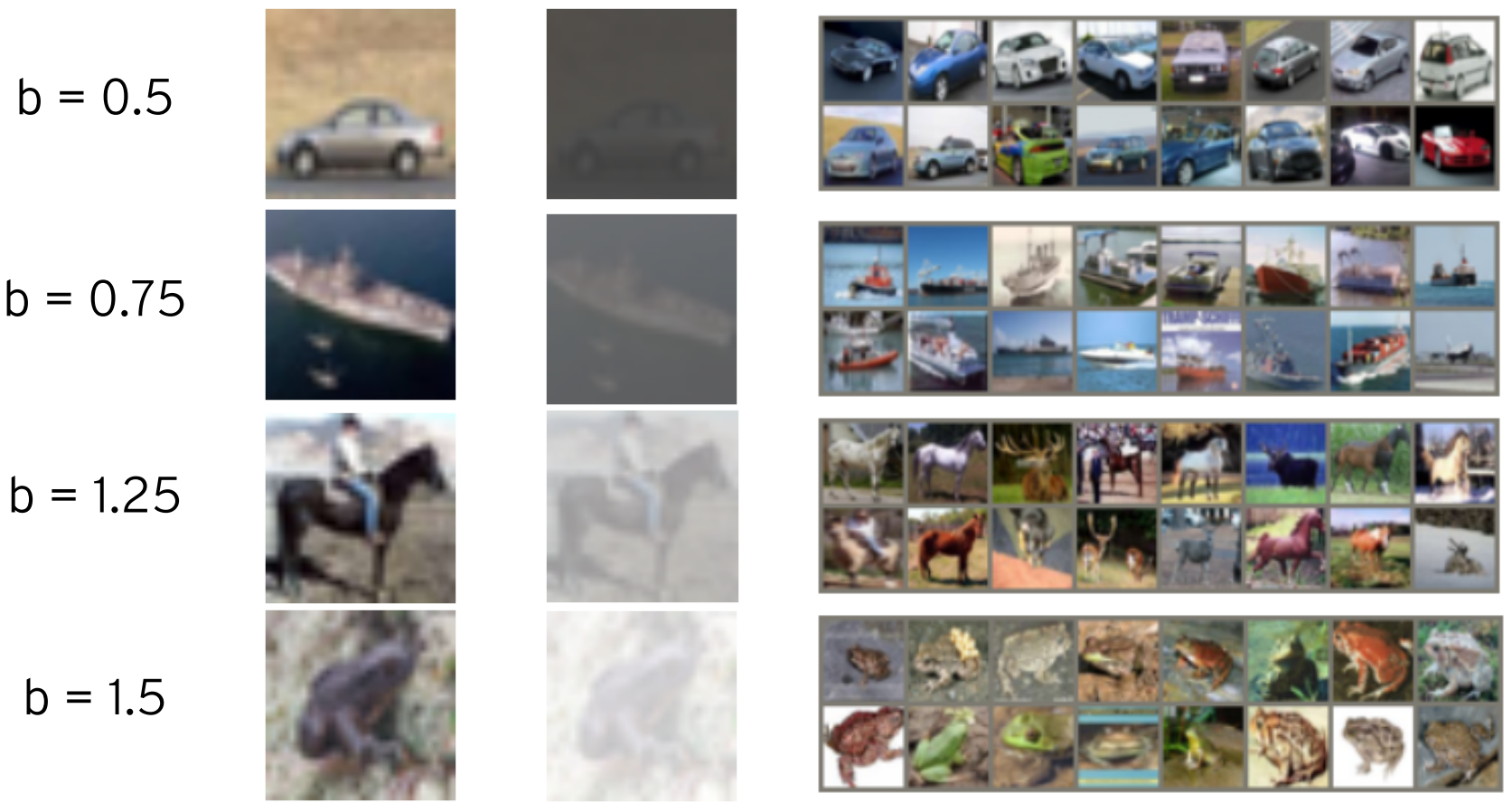
Appendix

Broken ResNet-18 with Triplet Loss: Effects of change in contrast



Appendix

Broken ResNet-18 with Triplet Loss: Effects of change in brightness



Appendix

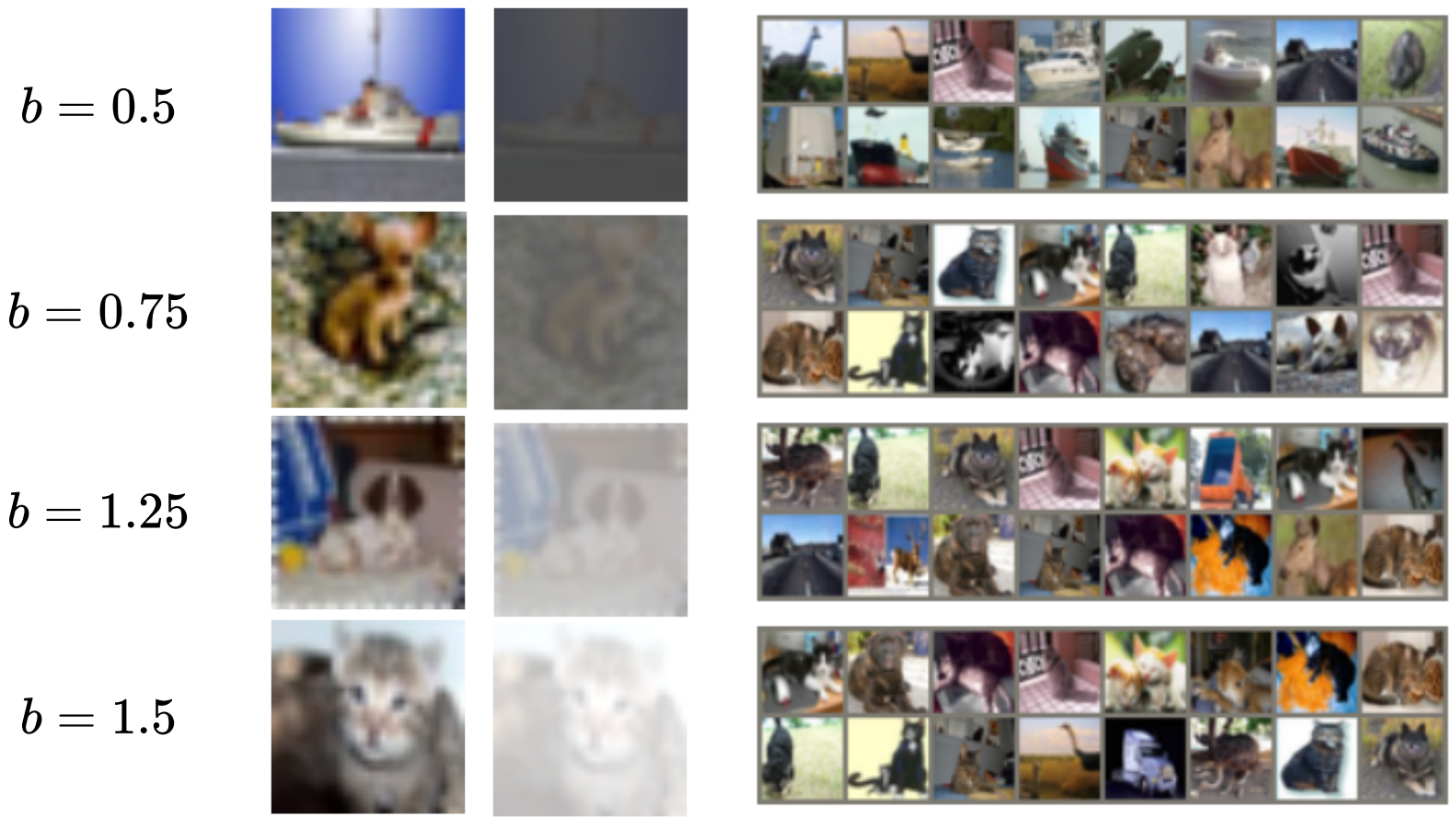
Broken ResNet-18 with Triplet Loss: Effects of change in γ-correction

Appendix

Broken ResNet-18 with N-Pair Loss: Effects of change in contrast

Appendix

Broken ResNet-18 with N-Pair Loss: Effects of change in brightness



Appendix

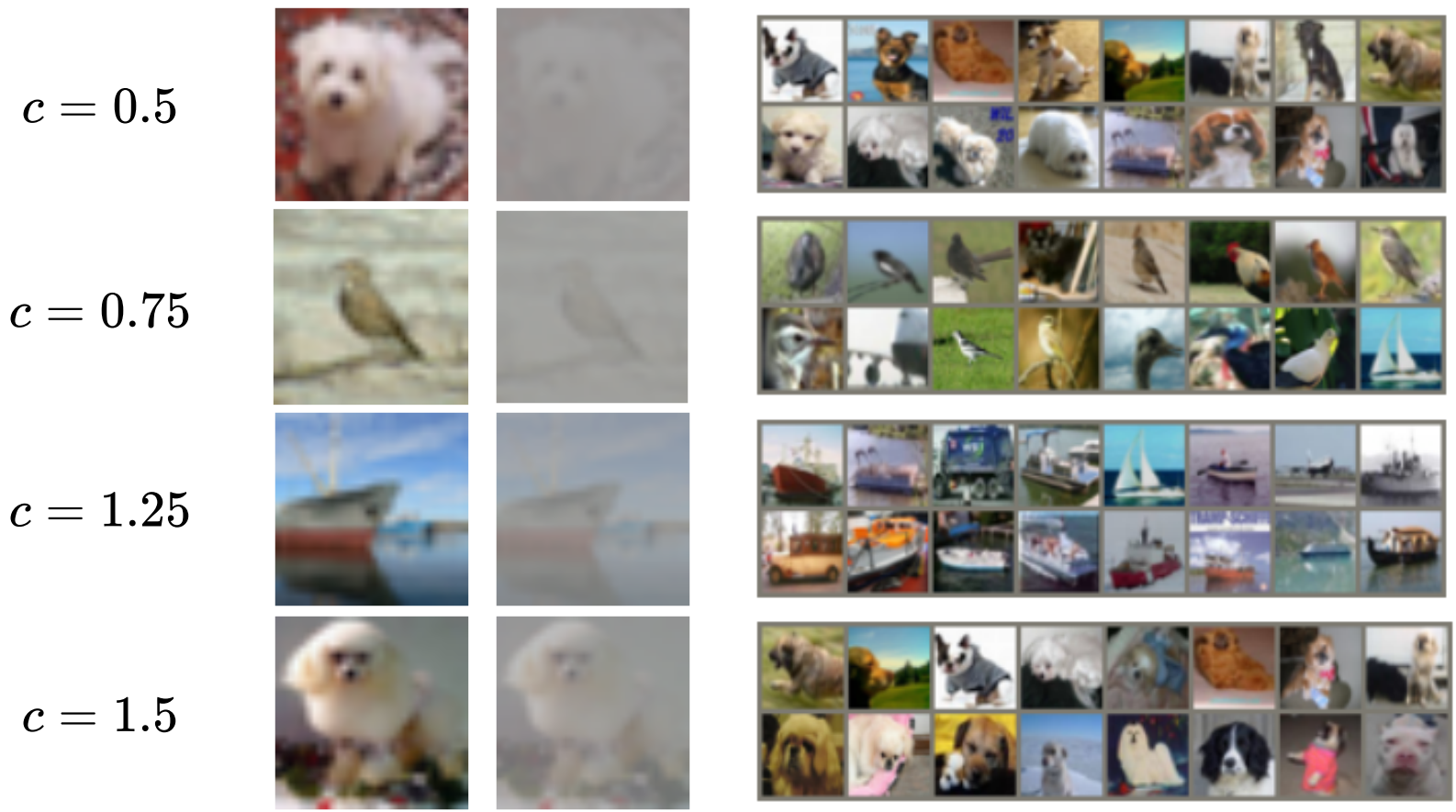
Broken ResNet-18 with N-Pair Loss: Effects of change in γ-correlation

Appendix

6-layer CNN with Triplet Loss: Effects of Adversial attack(FGSM)

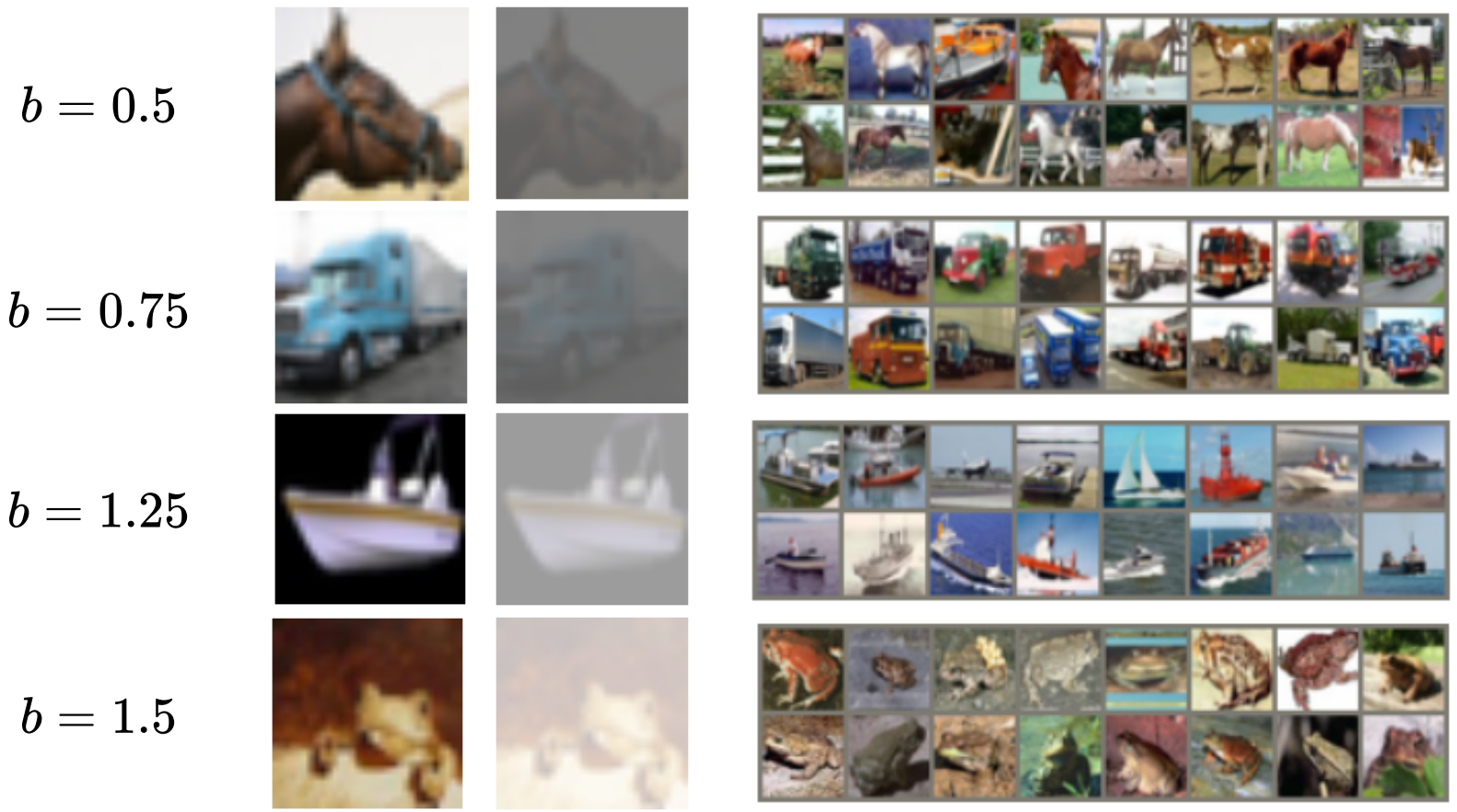
Appendix

6-layer CNN with Triplet Loss: Effects of contrast variation



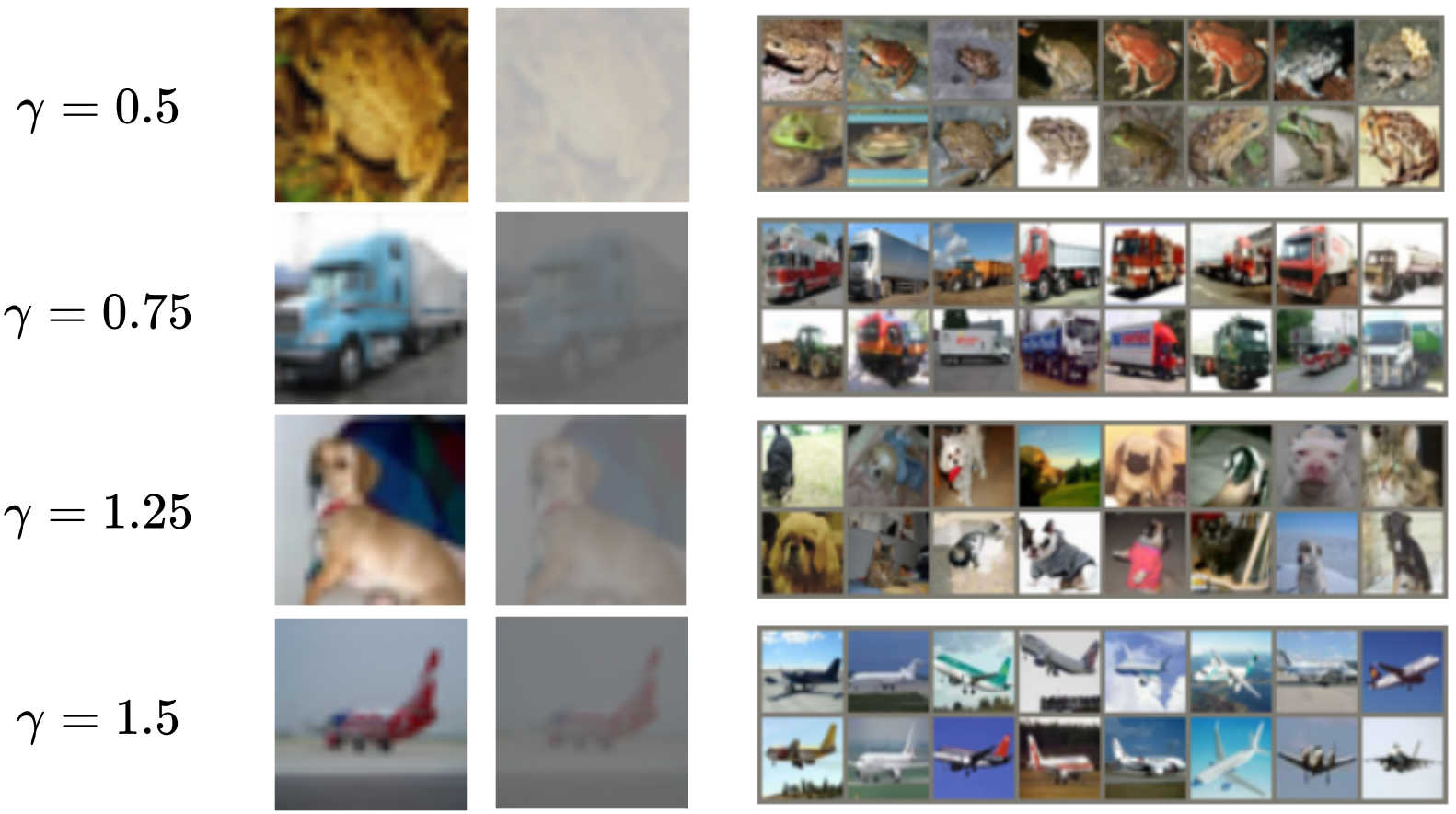
Appendix

6-layer CNN with Triplet Loss: Effects of brightness variation



Appendix

6-layer CNN with Triplet Loss: Effects of changing γ-correlation



Appendix

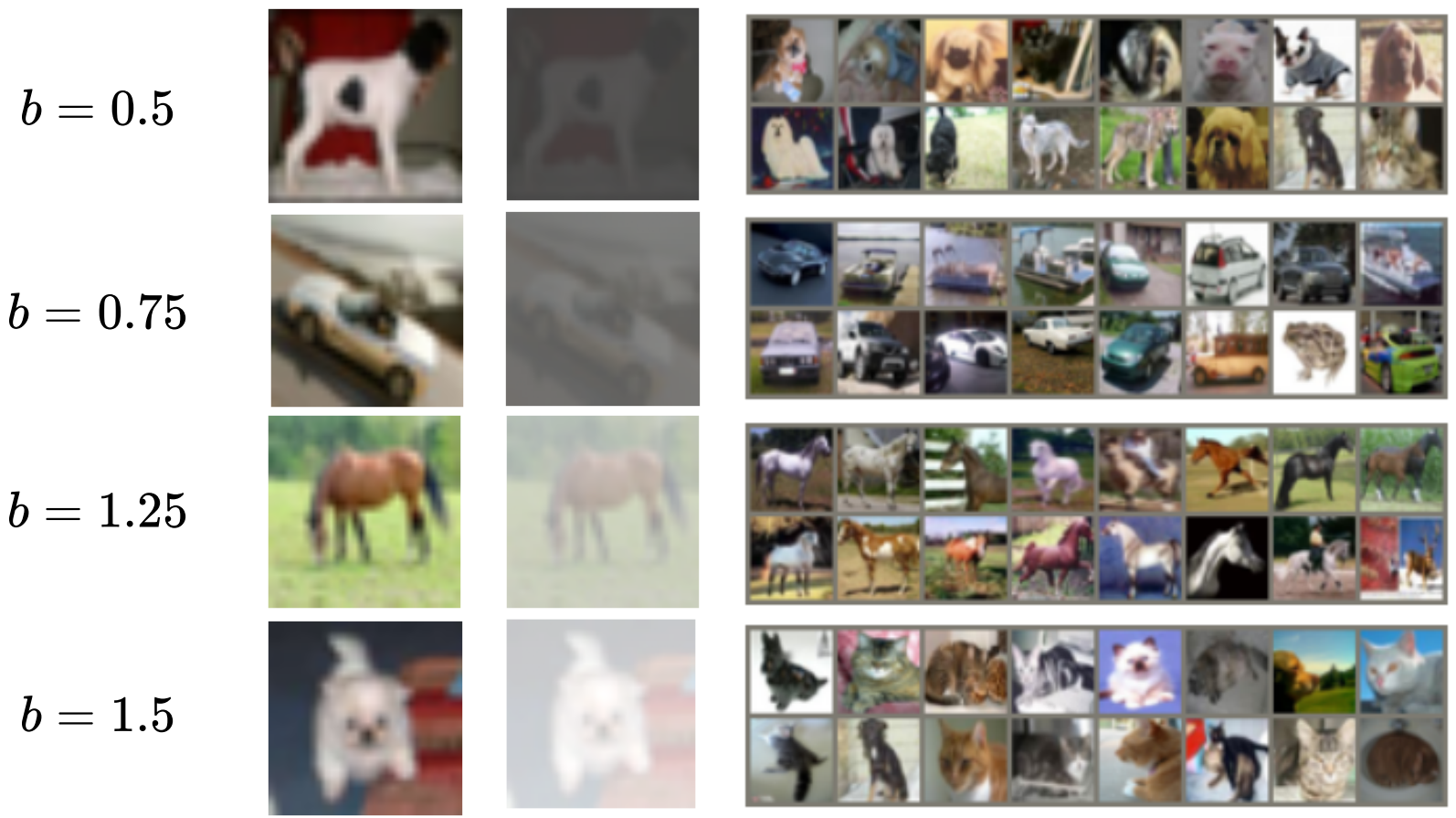
6-layer CNN with N-Pair Loss: Effects of Adversial attack(FGSM)

Appendix

6-layer CNN with N-Pair Loss: Effects of change in contrast

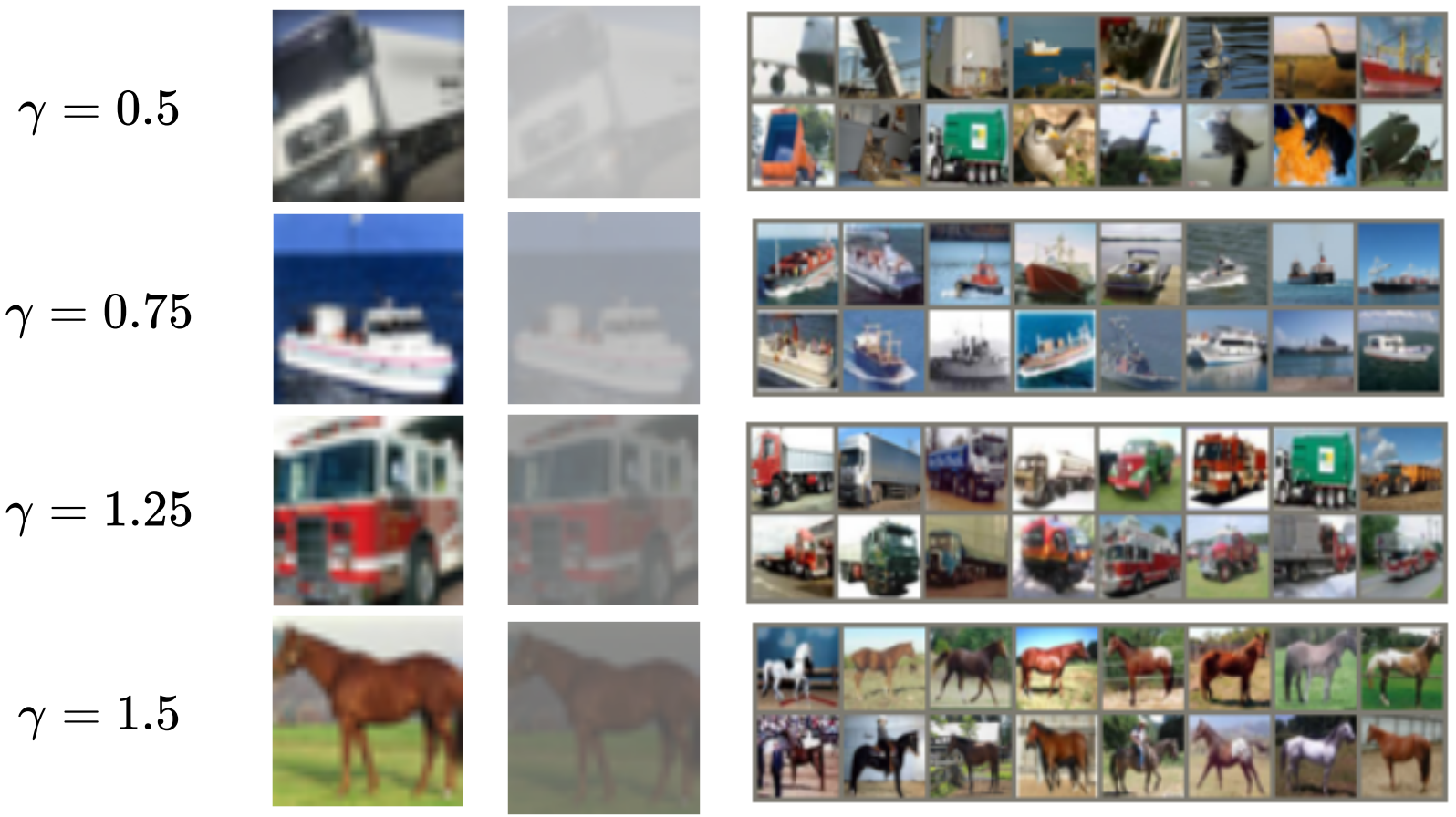
Appendix

6-layer CNN with N-Pair Loss: Effects of change in brightness



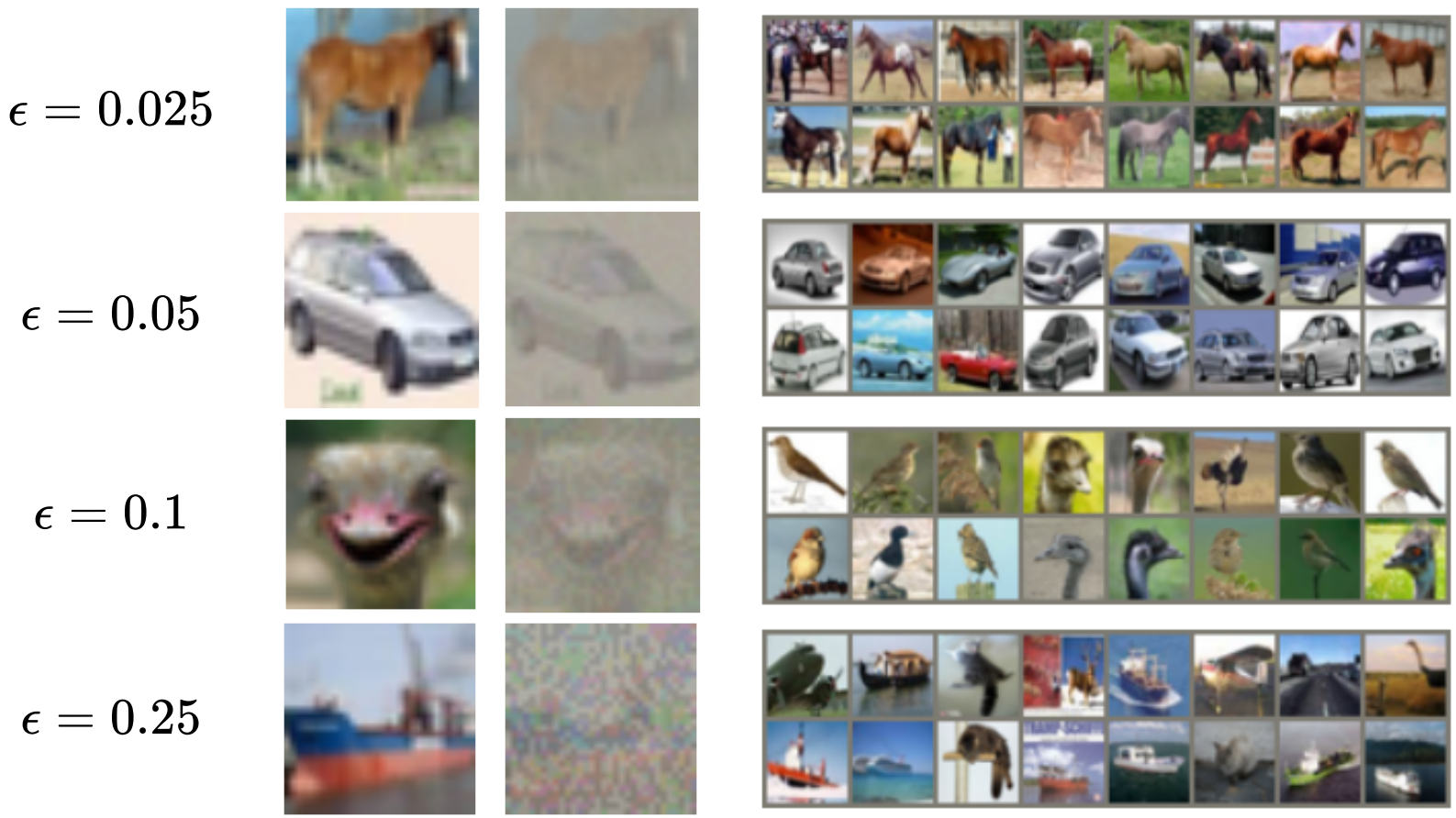
Appendix

6-layer CNN with N-Pair Loss: Effects of change in γ-correlation



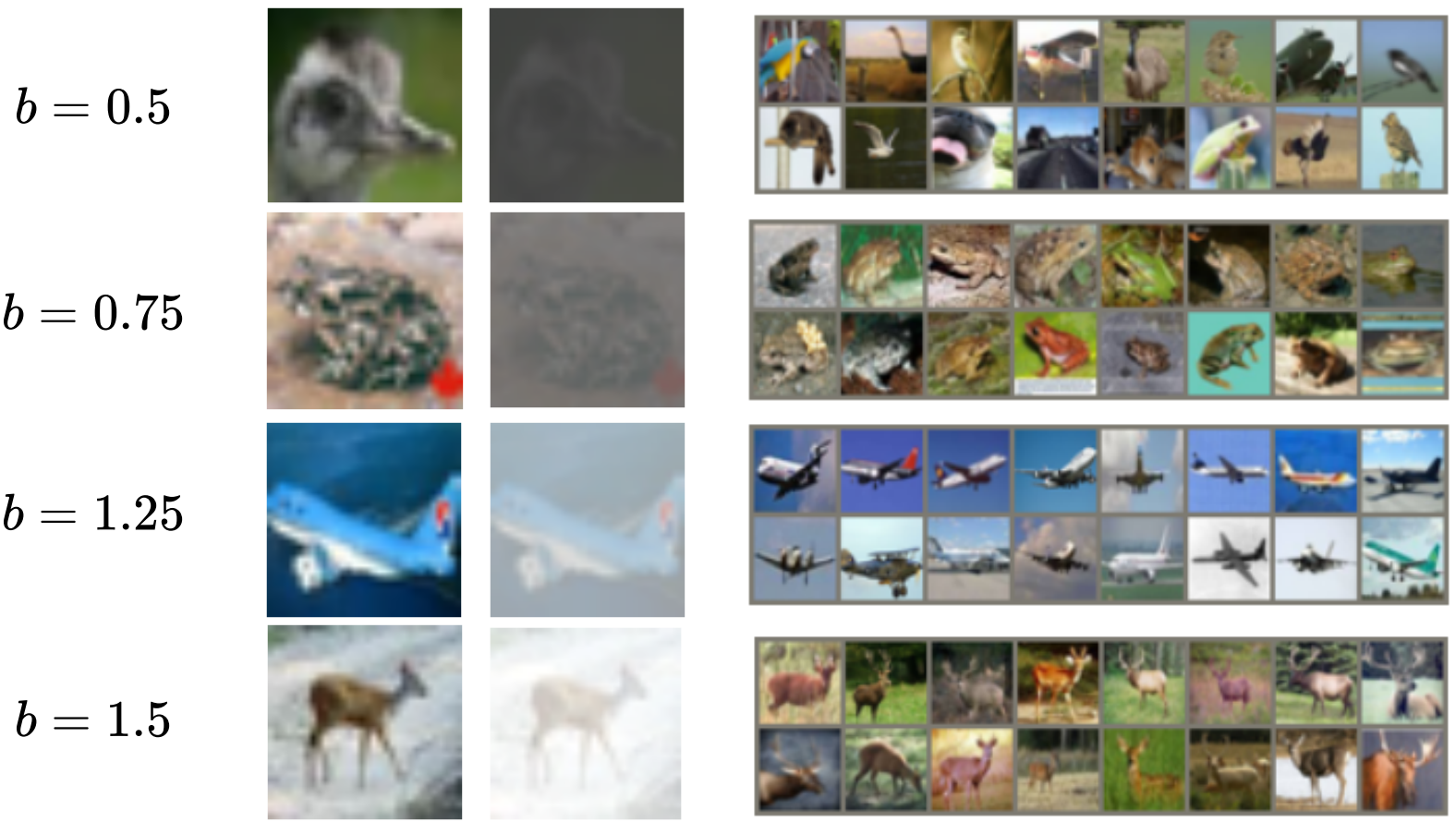
Appendix

HistNet with Triplet Loss: Effects of Adversial attack(FGSM)



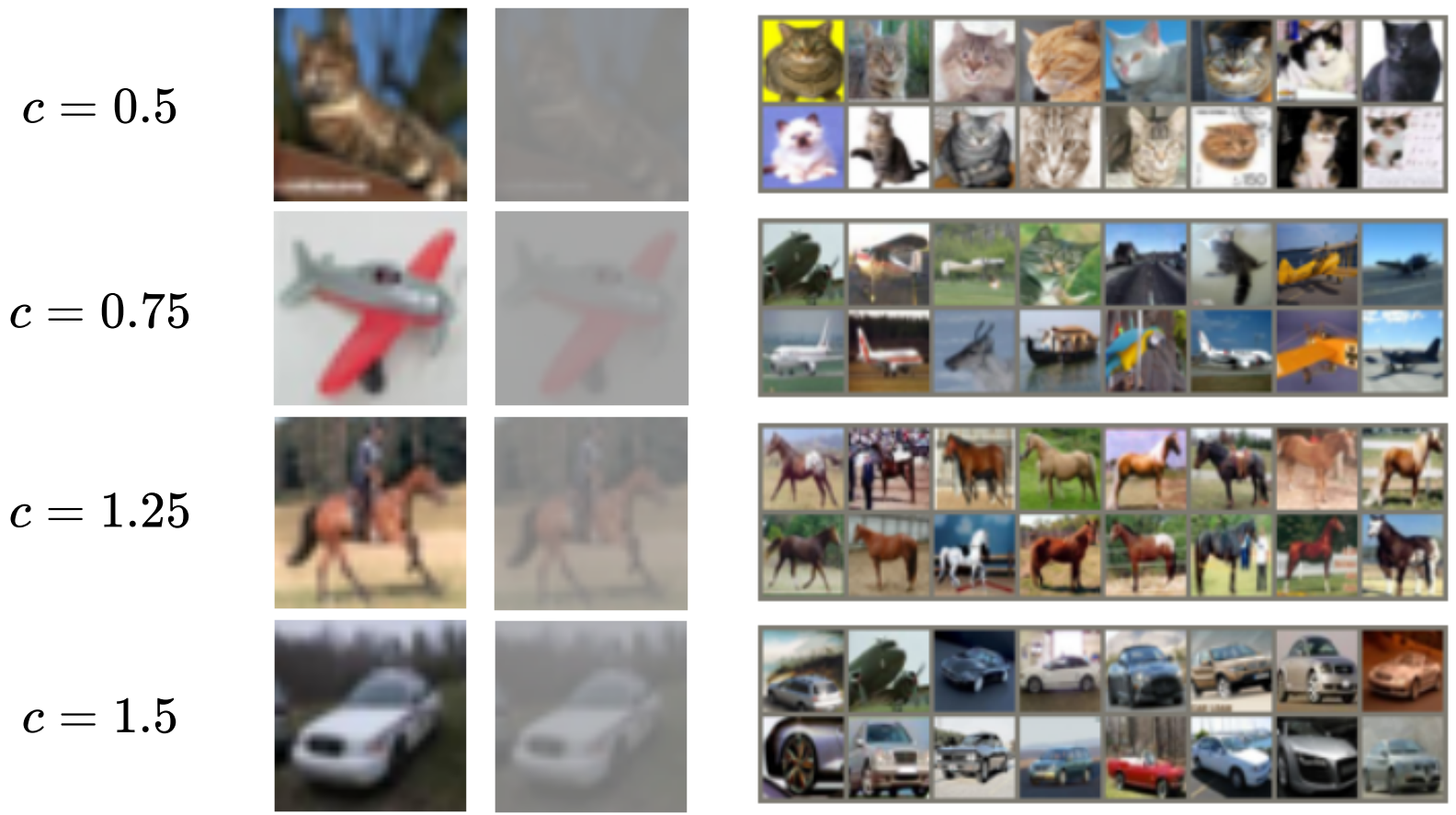
Appendix

HistNet with Triplet Loss: Effects of changing brightness



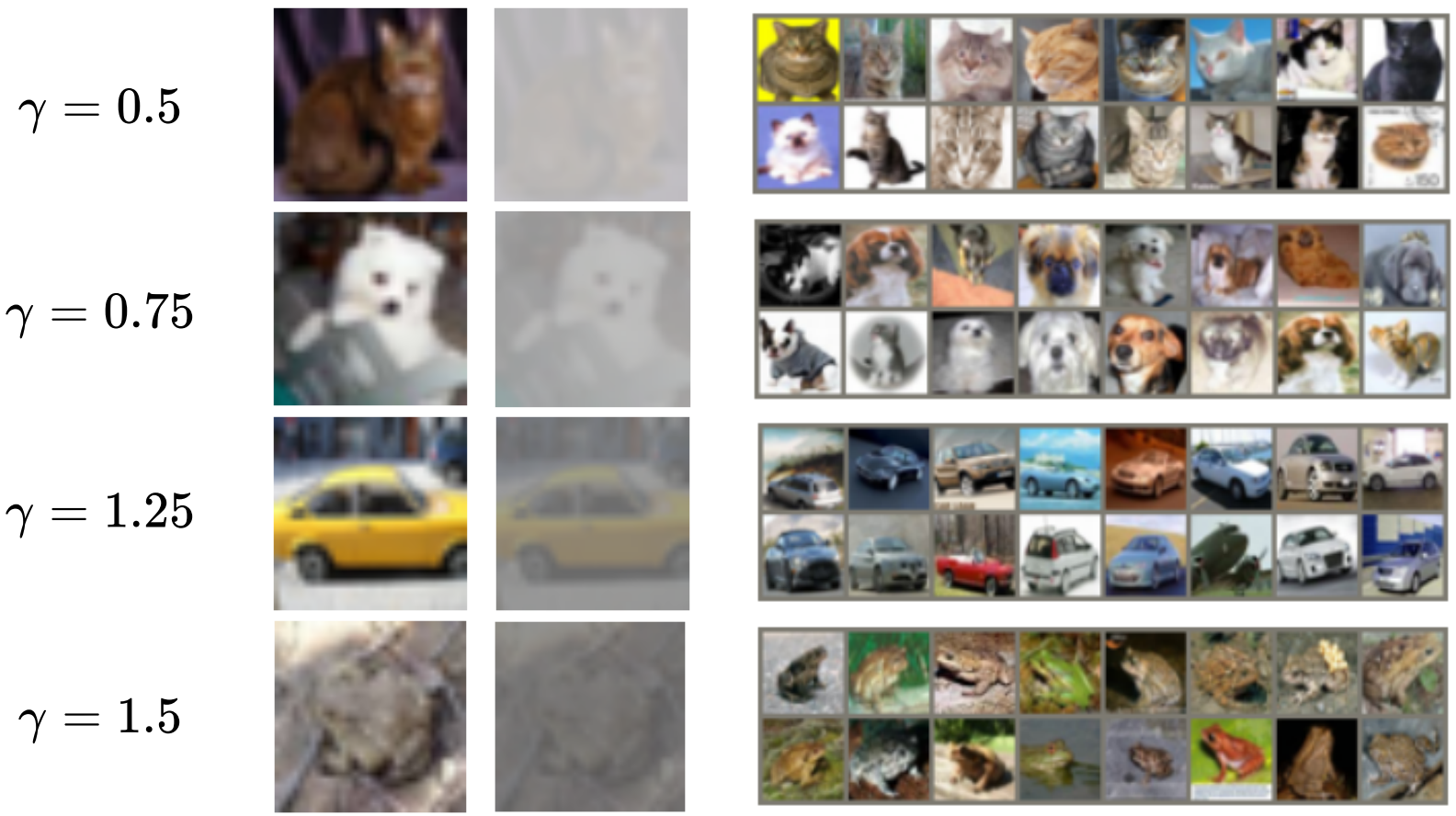
Appendix

HistNet with Triplet Loss: Effects of changing contrast



Appendix

HistNet with Triplet Loss: Effects of changing γ-correlation

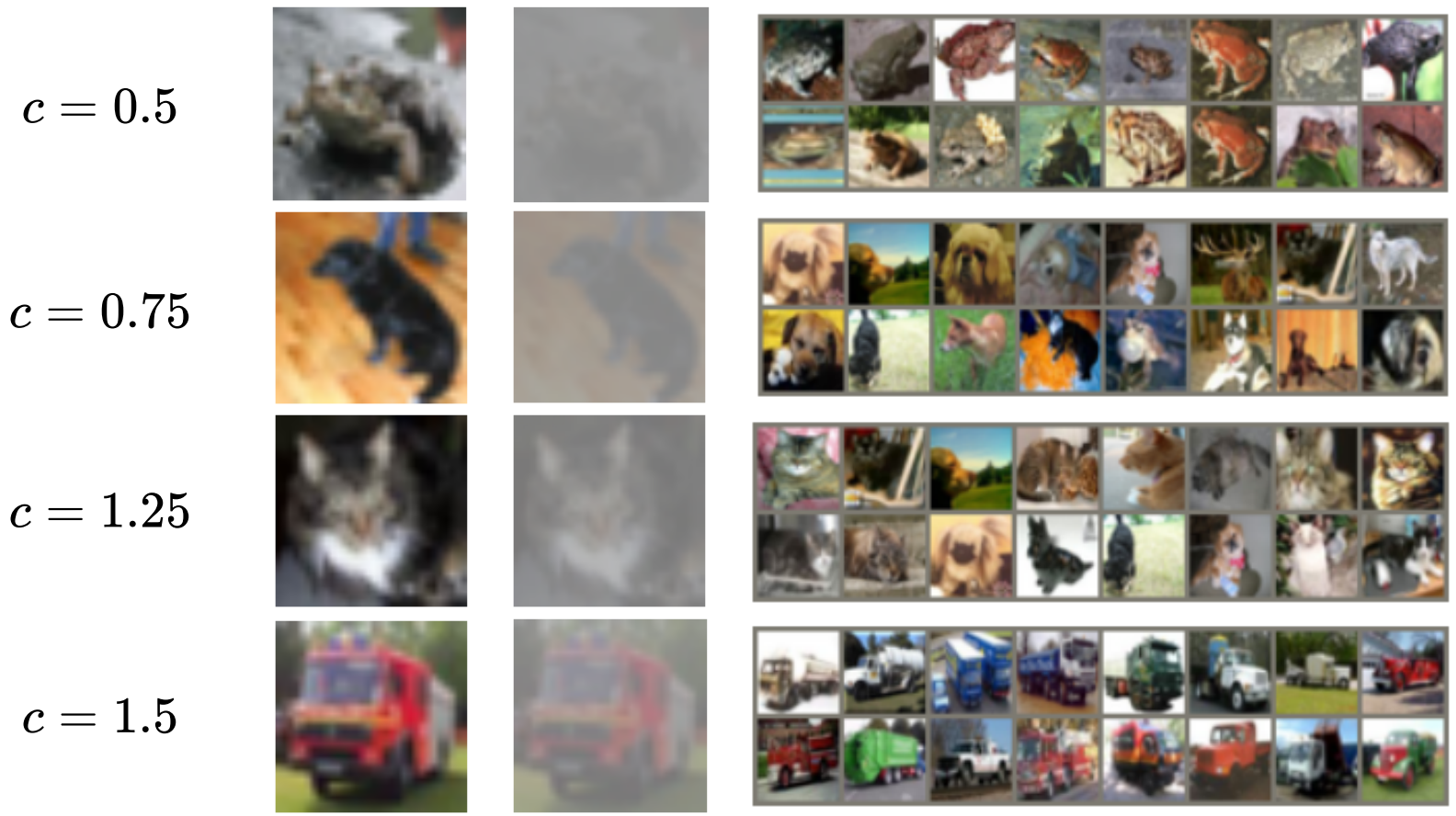


Appendix

HistNet with N-Pair Loss: Effects of Adversial attack(FGSM)

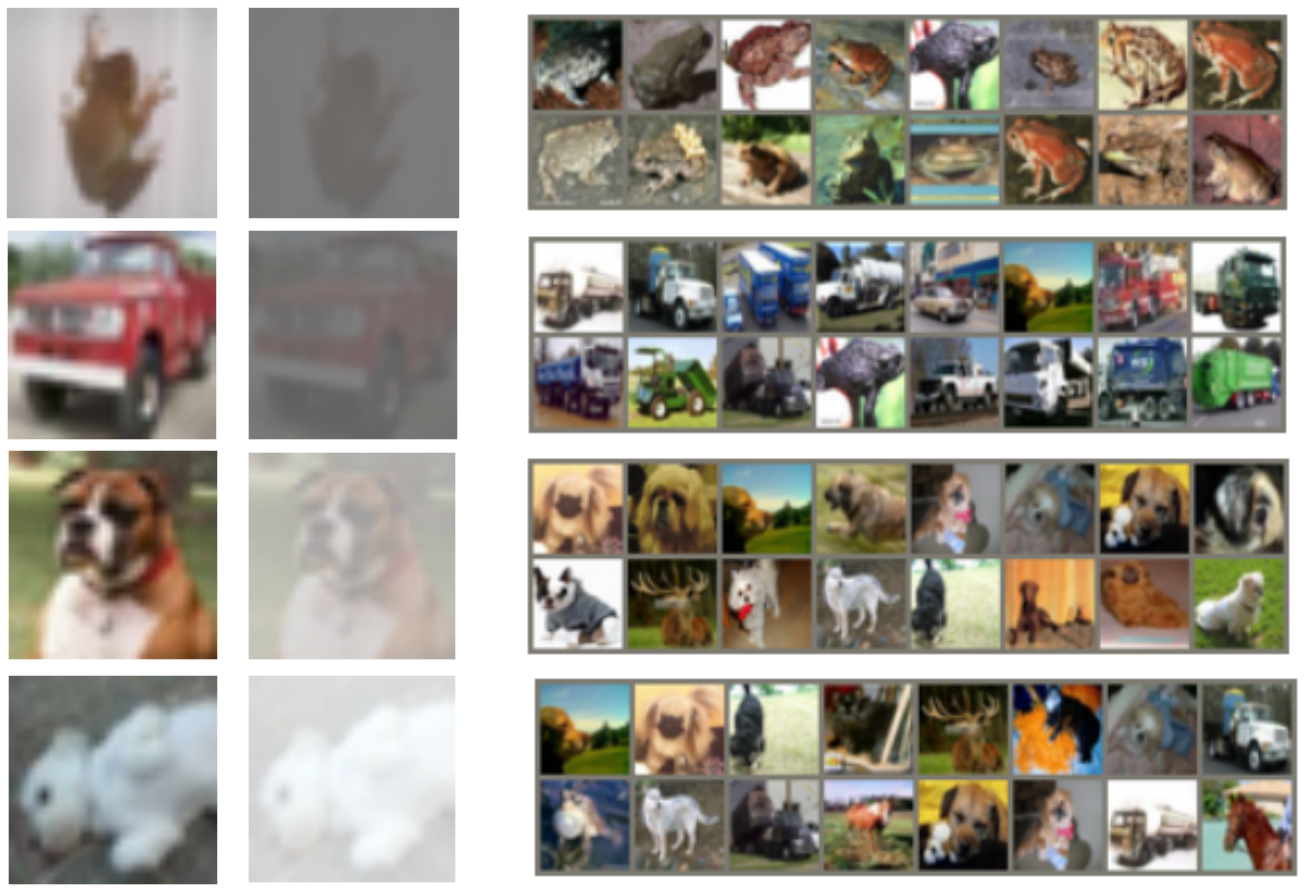
Appendix

HistNet with N-Pair Loss: Effects of changing contrast



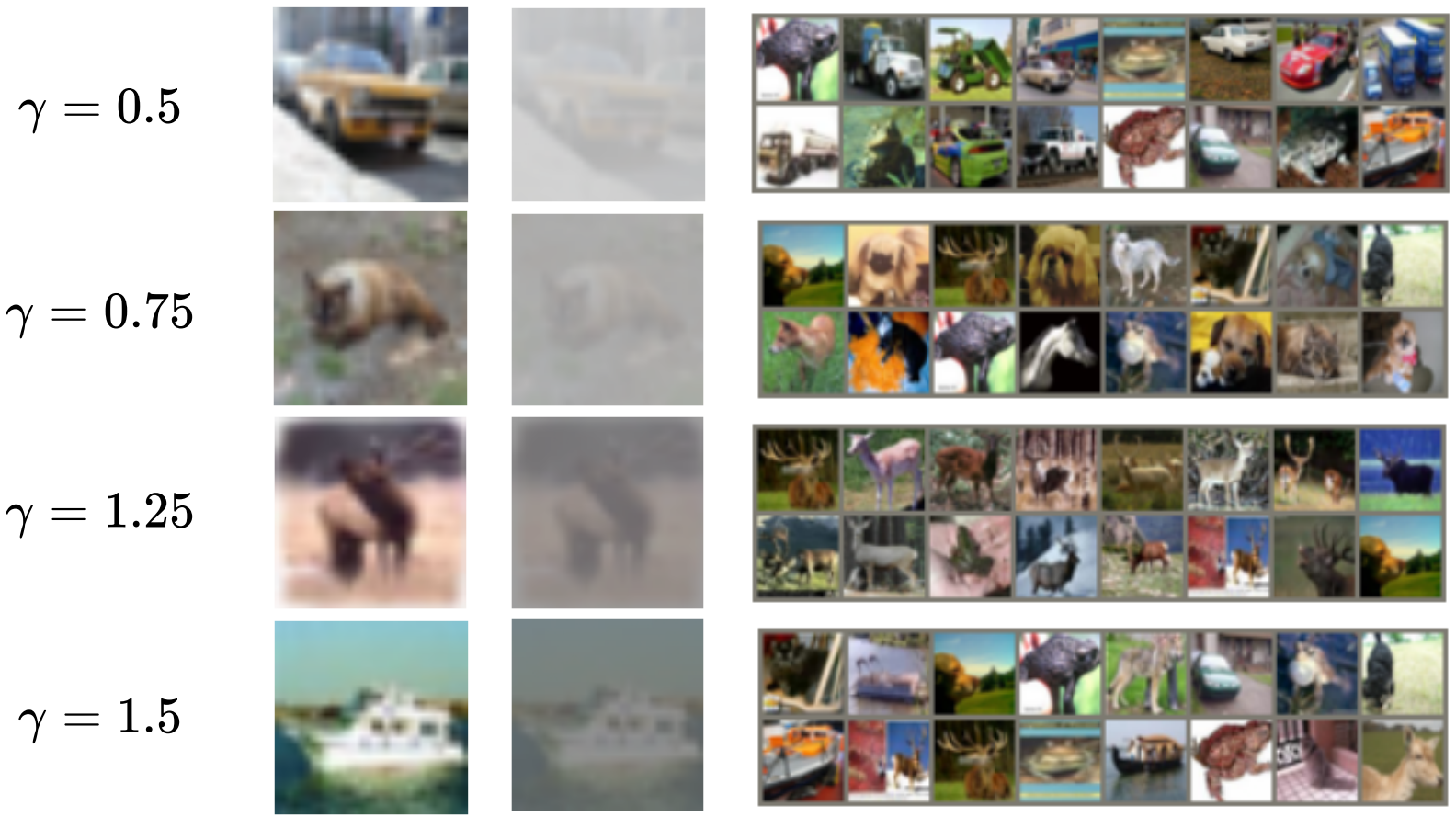
Appendix

HistNet with N-Pair Loss: Effects of changing brightness



Appendix

HistNet with N-Pair Loss: Effects of change in γ-correlation

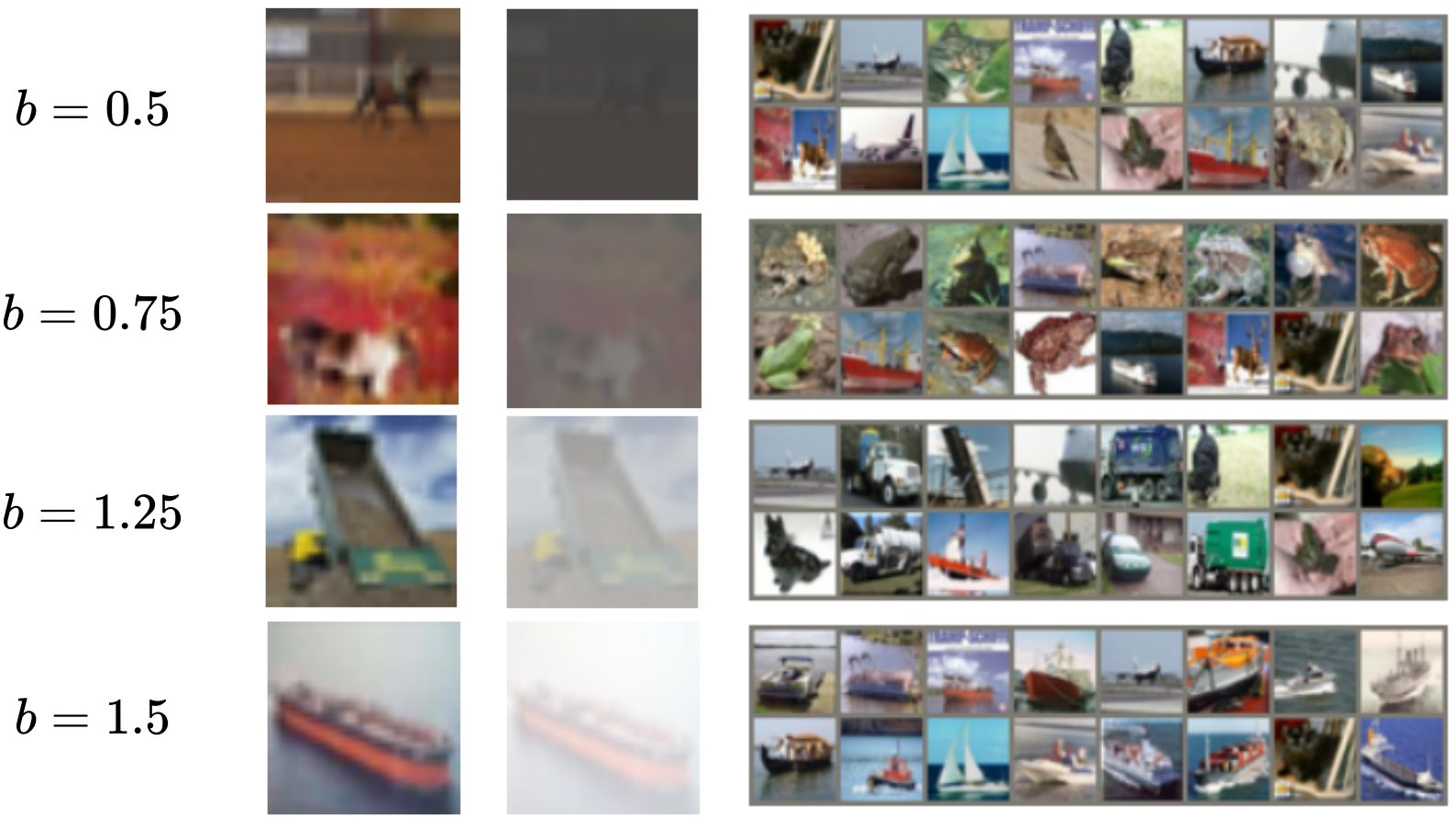


Appendix

FisherNet with Triplet Loss: Effects of Adversial attack(FGSM)

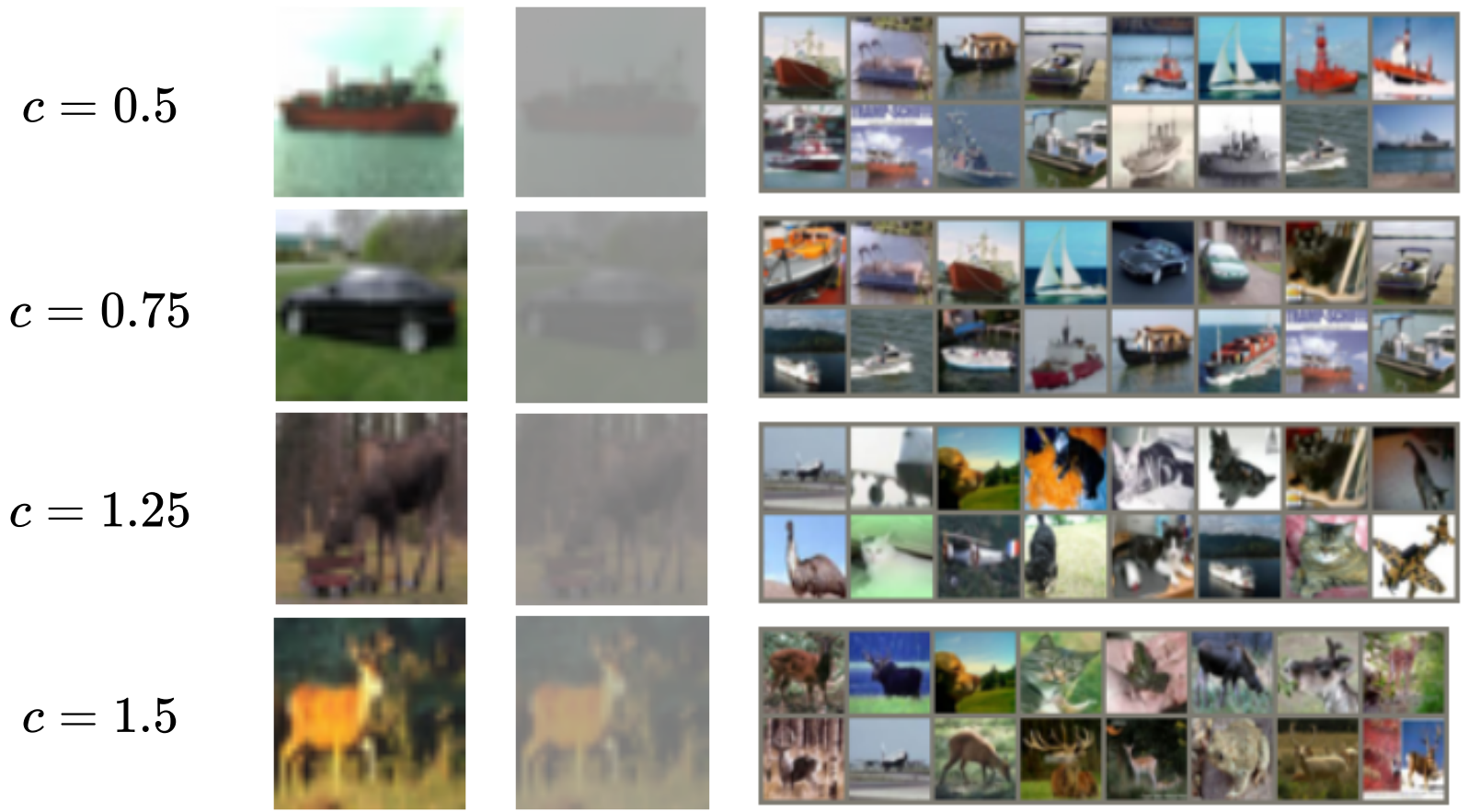
Appendix

FisherNet with Triplet Loss: Effects of changing brightness



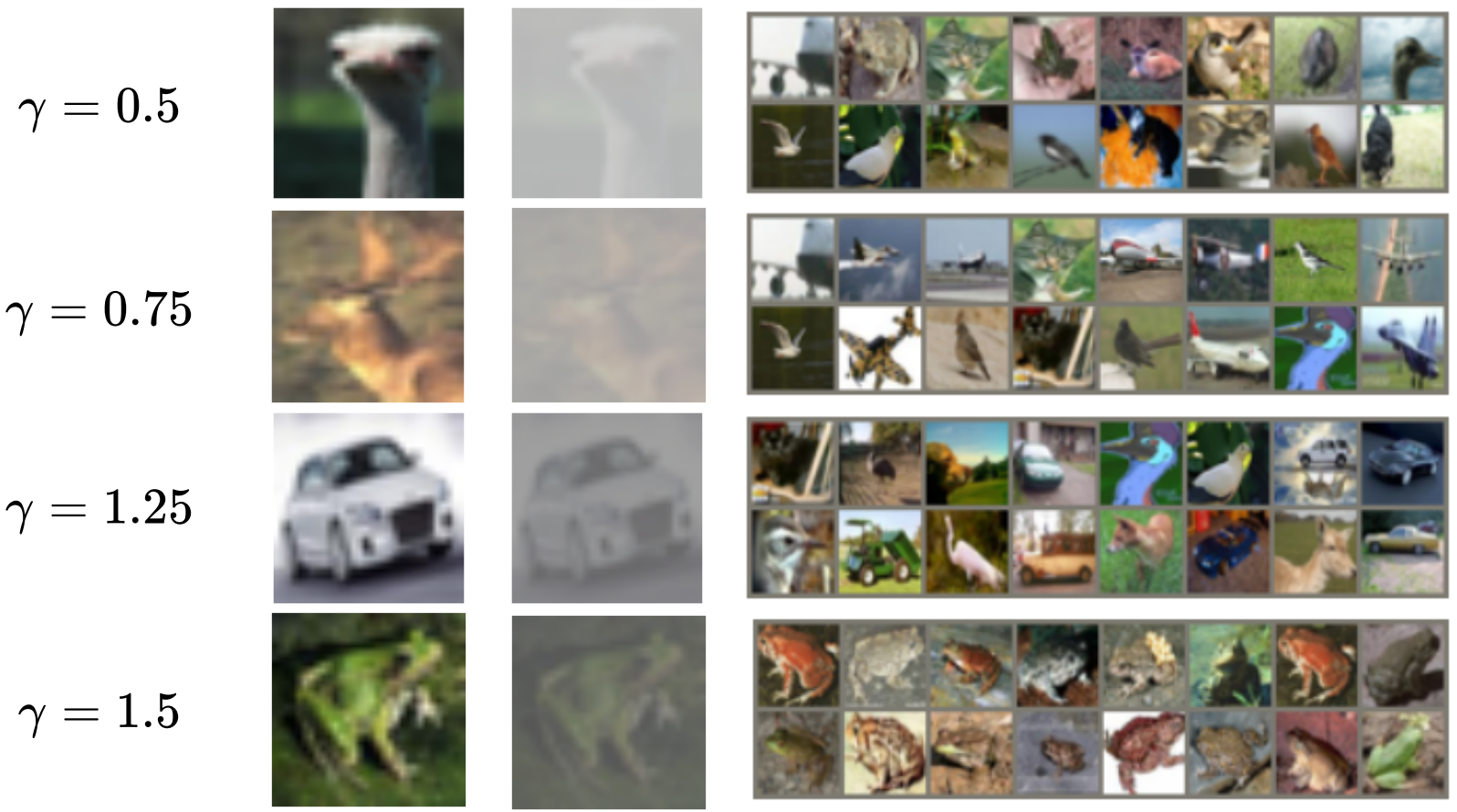
Appendix

FisherNet with Triplet Loss: Effects of changing contrast



Appendix

FisherNet with Triplet Loss: Effects of changing γ-correction

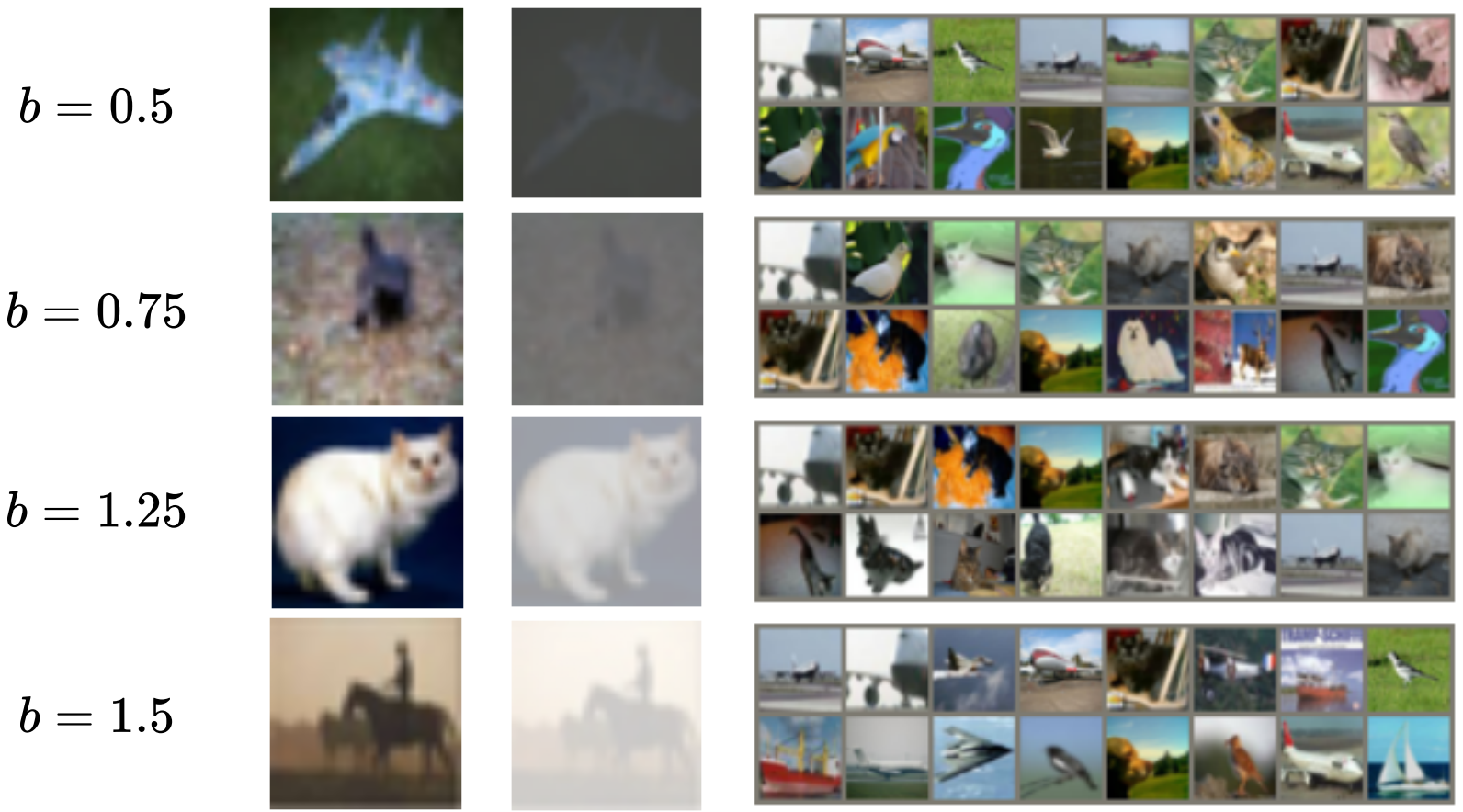


Appendix

FisherNet with N-Pair Loss: Effects of Adversial attack(FGSM)

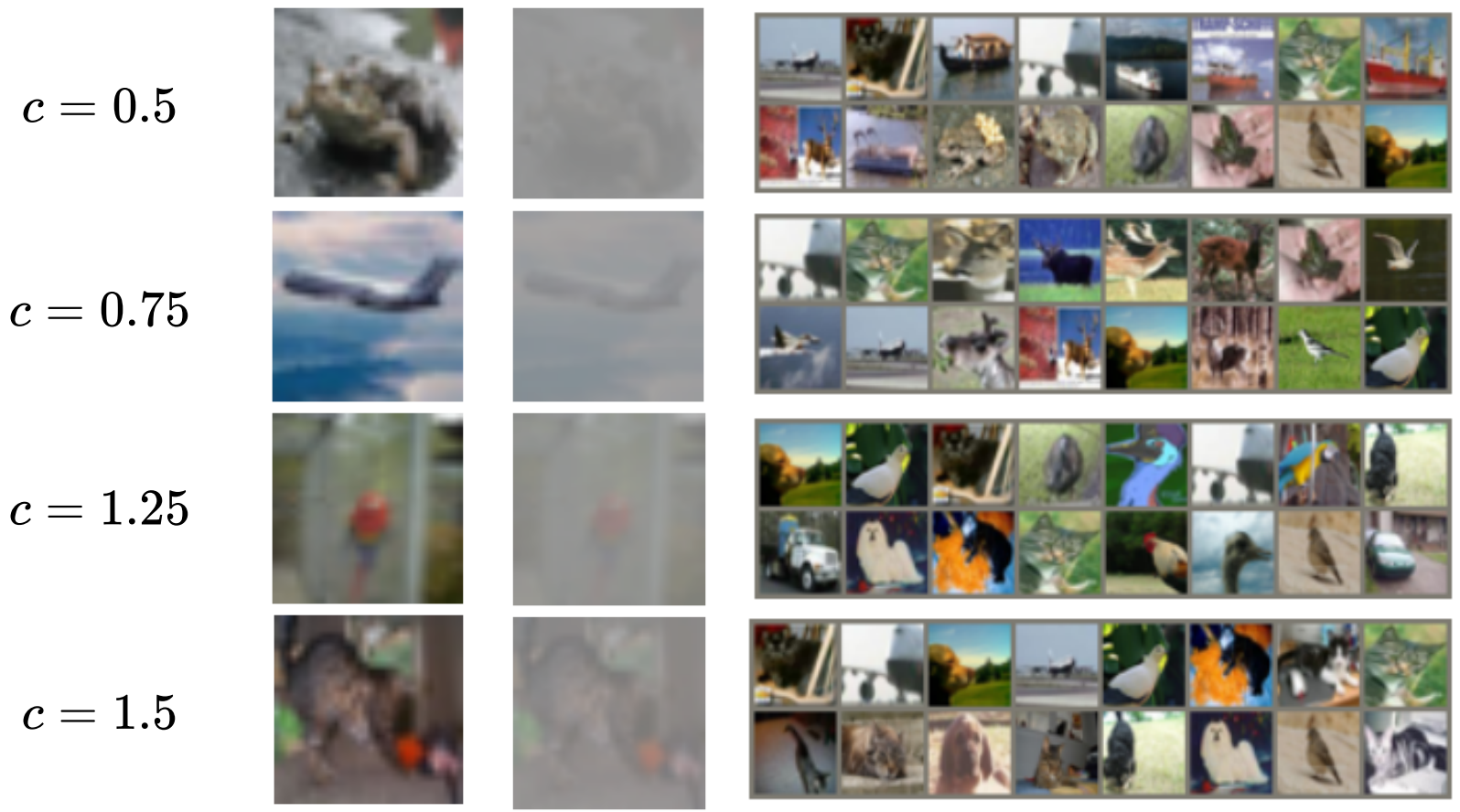
Appendix

FisherNet with N-Pair Loss: Effects of changing brightness



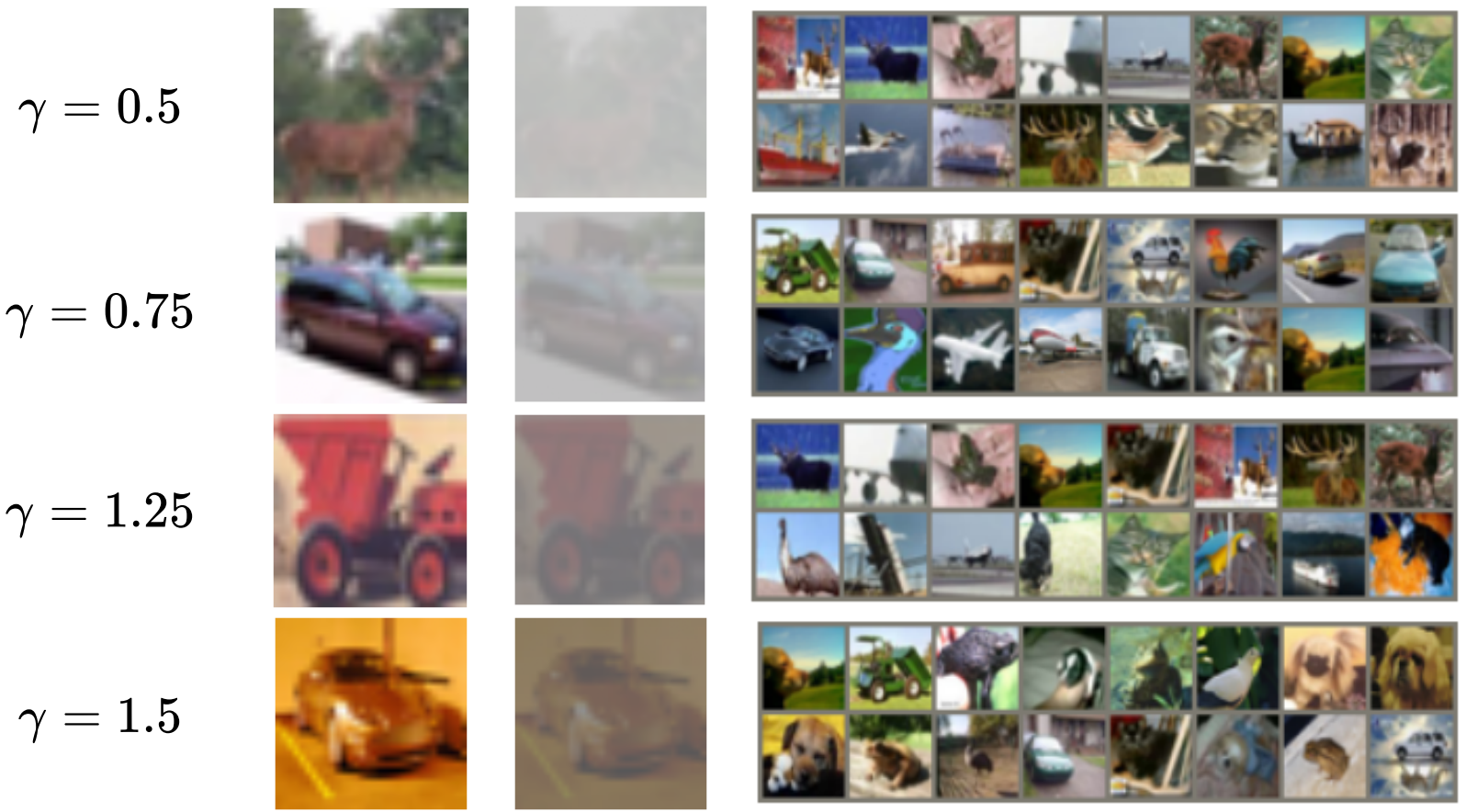
Appendix

FisherNet with Triplet Loss: Effects of changing contrast



Appendix

FisherNet with N-Pair Loss: Effects of changing γ-correction



Everybody is a genius. But if you judge a fish by its ability to climb a tree, it will live its whole life believing that it is stupid

-Albert Einsten

Shrisudhan
me17b122@smail.iitm.ac.in

SysDL Project
CS 6886