Mixup : Beyond Empirical Risk Minimization

AhmedElmogtaba Abdelmoniem Ali

Introduction

Issues with DNNs

- Large deep neural networks are powerful, but exhibit undesirable behaviors such as memorization and sensitivity to adversarial examples.

Issues with DNNs

These DNNs share two things in common :

1- They are trained to minimize their average error over the training data , which is called Empirical Risk Minimization (**ERM**) principle.

2- The size of these neural networks scales linearly with the number of training examples. And that does not guarantee the convergence of **ERM**.

======>

ERM allows large networks to memorize instead of generalize from training data even in the presence of strong regularization.

What is the Solution ?

Data Augmentation

Data Augmentation is to simply train on similar but different examples to the training data .

Advantages :

- Leads to an improve on generalization .

Disadvantages :

- The procedure is **dataset-dependent** , so it requires the use of expert knowledge.
- Data Augmentation assumes that the examples in the vicinity share the same class so it does not model the vicinity relation across examples of different classes.

Motivation ?

Try to come up with a simple data-agnostic data augmentation routine.

Idea

Mixup

$$\tilde{x} = \lambda x_i + (1 - \lambda) x_j$$

where x_i, x_j are raw input vectors

$$\tilde{y} = \lambda y_i + (1 - \lambda) y_i, \qquad \text{where } y_i,$$

y_j are one-hot label encodings

Formally , you take two data points and you mix them using the lambda mixing factor and you take the two corresponding labels and you mix them accordingly as well and that will give you the label.

”Mixup” extends the training distribution by incorporating the knowledge that linear interpolations of feature vectors should lead to linear interpolations of the associated targets.

$$\tilde{x} = \lambda x_i + (1 - \lambda) x_j,$$
$$\tilde{y} = \lambda y_i + (1 - \lambda) y_j,$$

$$\lambda \sim \mathrm{Beta}(\alpha, \alpha)$$
$$\alpha \in (0, \infty)$$

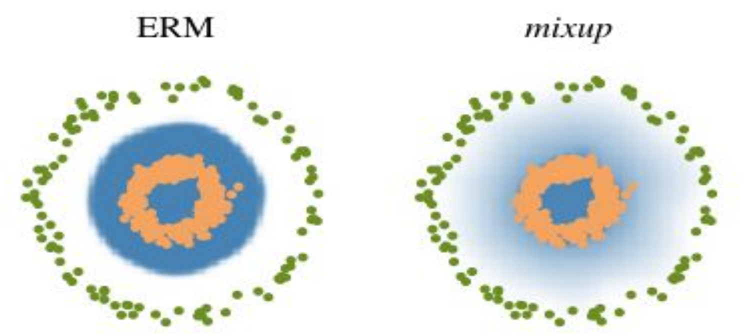
What is mixup doing?

The mixup distribution is a form of data augmentation that encourages the model to behave linearly in-between training examples.They argued that this linear behaviour reduces the amount of undesirable oscillations when predicting outside the training examples i.e (better generalization) .

Findings & Investigations

They showcase what mixup can do , so in a classic model as shown in the adjacent picture you have orange and green data points , and blue is basically where the classifier believes it's class one.

There is a hard border , and this border is in itself a problem because if you think of an adversarial examples all they have to do is get over that one inch and the classifier is already super sure it's the orange class. Whereas if you use mixup you boarder is much much more fuzzy.

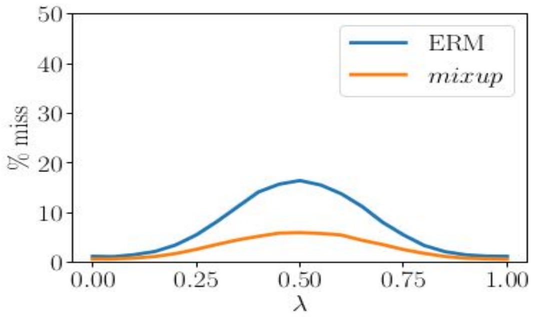


(b) Effect of *mixup* ($\alpha = 1$) on a toy problem. Green: Class 0. Orange: Class 1. Blue shading indicates $p(y = 1|x)$.

Findings & Investigations

They measure the prediction error of in between data , and what it means is they say that the prediction is counted as a miss if it doesn't belong to y_i or y_j , so basically you look at what the classifier says in between the two data points so you just interpolate the two data points and just measure what the classifier says , whenever the classifier either says y_i

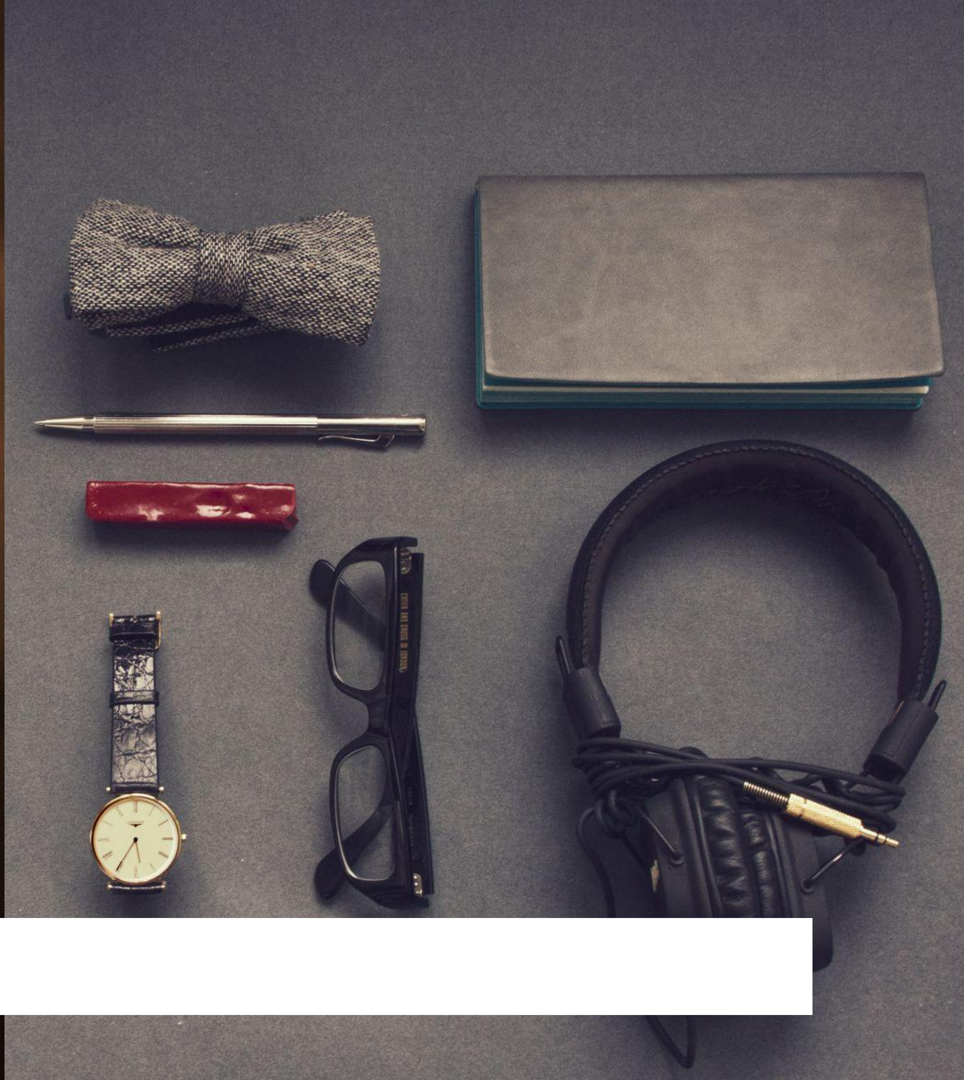
Or y_j you counted as correct and you counted as incorrect if it says something else.



(a) Prediction errors in-between training data. Evaluated at $x = \lambda x_i + (1-\lambda)x_j$, a prediction is counted as a "miss" if it does not belong to $\{y_i, y_j\}$. The model trained with *mixup* has fewer misses.

Experiments

- ImageNet Classification.
- CIFAR 10 and CIFAR 100.
- Speech data.
- Memorization of Corrupted labels.
- Robustness to adversarial Examples.
- Tabular data.
- Stabilization of Generative Adversarial networks.

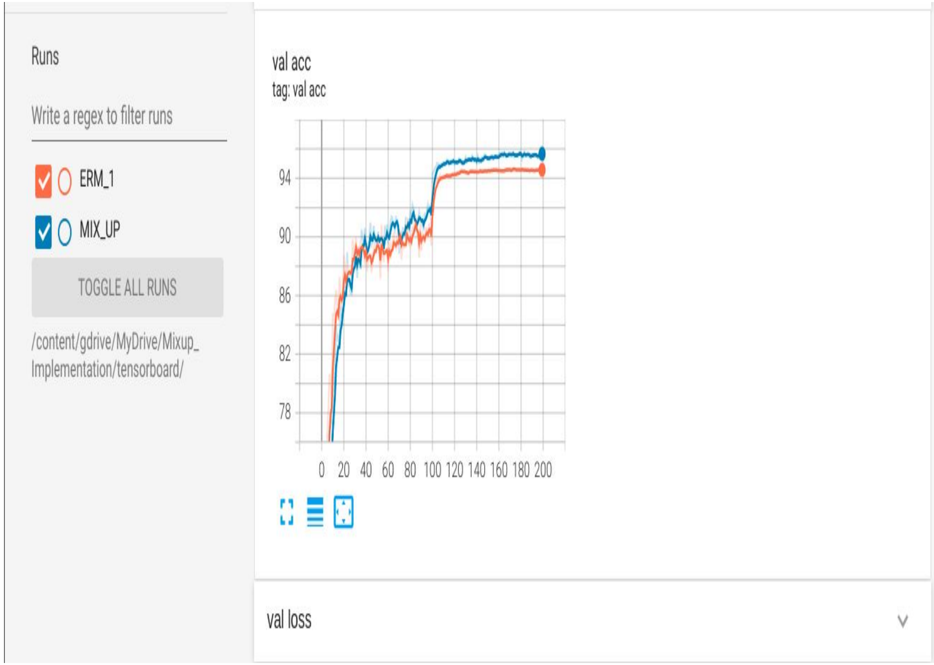


My Experiments

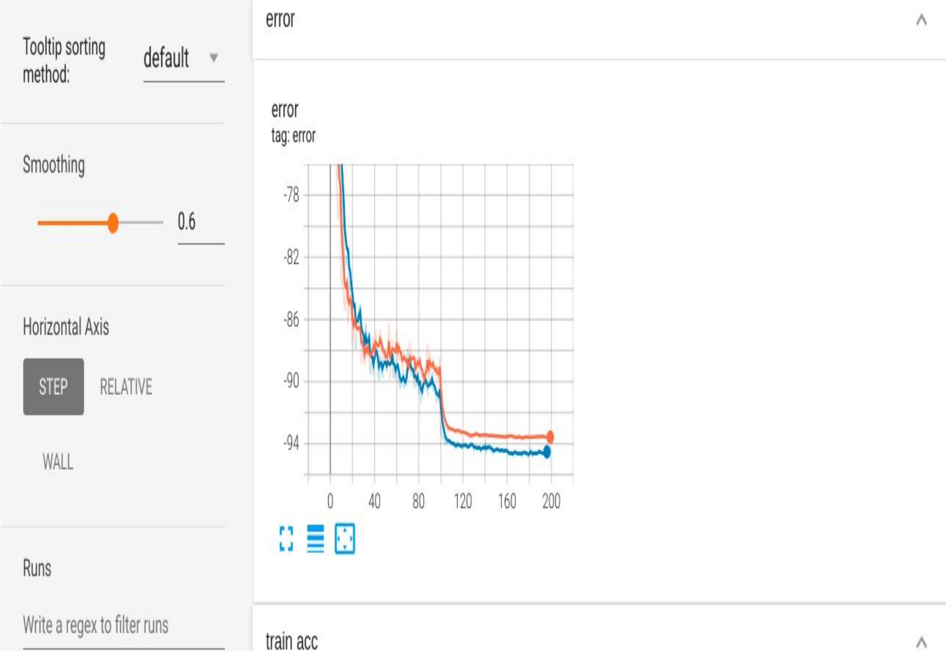
First Experiment :

- I conducted image classification experiment on CIFAR-10 dataset to evaluate the generalization performance of mixup
- I compared ERM and mixup training for: PreAct ResNet-18
- I got results similar to the paper , validation accuracy **94.63** for ERM And **95.79** for Mixup.

Graphs



Runs
Write a regex to filter runs
ERM_1
MIX_UP
TOGGLE ALL RUNS
/content/gdrive/MyDrive/Mixup_Implementation/tensorboard/
val acc
tag: val acc
94
90
86
82
78
0 20 40 60 80 100 120 140 160 180 200
val loss



Tooltip sorting method:
default
Smoothing
0.6
Horizontal Axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
error
error
tag: error
-78
-82
-86
-90
-94
0 40 80 120 160 200
train acc

Second Experiment : (Robustness to Adversarial Examples)

To assess the robustness of mixup model to adversarial examples, I used three ResNet18 models, two of them trained using ERM on CIFAR-10, and the third trained using mixup. In the first set of experiments, I studied the robustness of one ERM model and the mixup model against white box attacks. That is, for each of the two models, we use the model itself to generate adversarial examples, using the Fast Gradient Sign Method (FGSM) method. In the second set of experiments, I evaluated robustness against black box attacks. That is, we use the first ERM model to produce adversarial examples using FGSM . Then, we test the robustness of the second ERM model and the mixup model to these examples.

- For the White box test I got **53.7** (ERM) and **49.25** (Mixup) .

- For Black box test I got **55.07 (**ERM**)** and **49.76 (**Mixup**) .**

Comparison

Note the Architecture is Resnet18

Experiment	My Results	Paper Results	Justification
CIFAR_10 classification (**Mixup**)	4.21	4.2	The seed
CIFAR_10 classification (**ERM**)	5.37	5.6	The seed
White Box Attack (**FGSM**) (**Mixup**) Black Box (**FGSM**) (**Mixup**)	49.25 49.76	75.2 46.0	They used different dataset (**Imagenet**) And different architecture (**Resnet101**) + the seed
White Box Attack(**FGSM**)(**ERM**) Black Box Attack(**FGSM**)(**ERM**)	53.7 55.07	90.7 57.0	They used different dataset (**Imagenet**) And different architecture (**Resnet101**) + the seed

Conclusion

The End