Computational resources: Exploratory experiments were conducted using a desktop machine with two Nvidia RTX A6000 GPUs. The extended experimental results in the appendix were obtained on a shared HPC system, where a total of 3500 GPU hours were used across the lifetime of the project (Nvidia A100 GPUs).

A PROOFS

Here we give proofs of the theorems stated in the main body of the paper. To prove our first theorem, Theorem 4.2, we require the following differential-geometric lemma.

Lemma A.1. *Let M be any compact, embedded, d-dimensional submanifold of $\mathbb{R}^n$, possibly with boundary and corners. Equip M with the Riemannian structure inherited from this embedding and let $\mathbb{P}$ be the normalised volume measure of M induced by its Riemannian structure. Then there exists $\kappa > 0$ and $C > 0$ such that*

$$\mathbb{P}[B(x,r)] \geq C\, r^d$$

for all $r \leq \kappa$ and $x \in M$, where $B(x,r)$ denotes the closed Euclidean ball around x of radius r.

Proof. Since the Riemannian structure on M is inherited from its embedding into $\mathbb{R}^n$, the embedding is bi-Lipschitz and hence there exists a constant c such that

$$\|x-y\|_2 \leq c^{-1} g(x,y)$$

for all $x, y \in M$, where $g(x,y)$ denotes the geodesic distance induced on M by the Riemannian structure. It follows that for all $r > 0$, one has $B_g(x,cr) \subset B(x,r)$, where $B_g(x,r)$ is the closed geodesic ball of radius r about x. Hence

$$\mathbb{P}[B(x,r)] \geq \mathbb{P}(B_g(x,cr)) \tag{11}$$

and it remains only to lower-bound $B_g(x,cr)$ for r sufficiently small. This we achieve by considering the geometry of M.

Given $x \in M$, let $\exp_x : T_xM \to M$ be the Riemannian exponential map and let $(x,0) \in T_xM$ denote the zero tangent vector. Recall that the *injectivity radius* of M is the largest number R for which the restriction of $\exp_x$ to the R-ball about zero in T_xM is a diffeomorphism onto its image for all $x \in M$. The injectivity radius exists and is strictly positive by compactness of M. Let $\kappa > 0$ be any number that is strictly smaller than $c^{-1}R$.

Let us now fix $x \in M$. Since $\exp_x$ is a radial isometry, it maps $B((x,0),r)$ onto $B_g(x,r)$ for any $r \leq \kappa$. Let $d\xi$ denote the standard Euclidean volume element in $B((x,0),r)$ and ω the volume form on M. Since $\exp_x|_{B((x,0),r)}$ is a diffeomorphism, there exists a nonvanishing, smooth function f on $B((x,0),r)$ for which $\exp_x^* \omega = f_x\, d\xi$. Letting c'_x denote the minimum of f_x, we then have

$$\mathbb{P}[B_g(x,r)] = \int_{B_g(x,r)} \omega = \int_{B((x,0),r)} \exp_x^* \omega \geq c'_x \int_{B((x,0),r)} d\xi = c'_x\, c'' r^d. \tag{12}$$

Here c'' is the usual scaling factor one sees in the volume of a Euclidean d-ball. Using compactness of M, we define $c' := \inf_{x \in M} c'_x > 0$ and $C := c'\, c''\, c^d$. It then follows from the estimates (11) and (12) that

$$\mathbb{P}[B(x,r)] \geq C\, r^d$$

for all $x \in M$ and $0 < r \leq \kappa$ as claimed. □

Proof of Theorem 4.2. The proof we give is derived from (Reznikov & Saff, 2016, Theorem 2.1), however we wish to be more precise with our constants than is the case in Reznikov & Saff (2016). We thus give a full proof here.

First, using the fact that supp($\mathbb{P}$) satisfies the hypotheses of Lemma A.1, fix $C > 0$ and $\kappa > 0$ for which $\mathbb{P}[B(x,r)] \geq C\, r^d$ for all $(x,r) \in M \times [0,\kappa]$, and define $\Phi(r) := C\, r^d$. This function Φ will play a key role in the proof.

Now fix $\epsilon > 0$, let $X_N := (x_1, \ldots, x_N)$ be a set of i.i.d. points sampled from $\mathbb{P}$, and let $\rho(X_N) := \sup_x \min_i \|x - x_i\|$ denote the covering radius of X_N. Suppose that $\rho(X_N) > 2t$ for some real number t, and let E_t be any maximal set of points in supp($\mathbb{P}$) whose distinct pairwise distances are all bounded below by t. Then we can find $x \in E_t$ such that $B(x,t) \cap X_N = \emptyset$. Indeed, by hypothesis on $\rho(X_N)$, there exists $y \in \mathrm{supp}(\mathbb{P})$ such that $X_N \cap B(y, 2t) = \emptyset$. There also exists $x \in E_t$ such that $\|x - y\| < t$, since otherwise we could add y to E_t and contradict the maximality of E_t. One then has $B(x,t) \subset B(y,2t)$, so that $B(x,t)$ does not intersect X_N.

It has thus been shown that

$$\mathbb{P}[\rho(X_N) > 2t] \leq \mathbb{P}[\exists x \in E_t : B(x,t) \cap X_N = \emptyset]. \tag{13}$$

Since $\mathbb{P}(B(x,t)) \geq \Phi(t)$ for all x, $(1 - \Phi(t))$ is an upper bound on the probability that a randomly selected x' will not lie within t of a given x. Letting $\#(E_t)$ denote the cardinality of E_t, the independence of the x_i therefore permits an upper bound

$$\mathbb{P}[\rho(X_N) > 2t] \leq \#(E_t)\,(1 - \Phi(t))^N. \tag{14}$$

The cardinality of E_t can be further bounded via

$$1 \geq \sum_{x \in E_t} \mathbb{P}[B(x,t)] \geq \#(E_t)\Phi(t), \tag{15}$$

so that one has

$$\mathbb{P}[\rho(X_N) > 2t] \leq \Phi(t)^{-1}(1 - \Phi(t))^N. \tag{16}$$

Using the invertibility of Φ, one now sets $\Phi(t) = \alpha \log(N) N^{-1}$, with $\alpha = (1 - \log_N(\epsilon))$. The McLaurin series for $\log(1-y)$ with $y = \alpha \log(N) N^{-1}$ reveals that

$$(1 - \alpha \log(N) N^{-1})^N \leq N^{-\alpha}, \tag{17}$$

so that the bound becomes

$$\mathbb{P}[\rho(X_N) > 2\Phi^{-1}(\alpha \log(N) N^{-1})] \leq \frac{1}{\alpha} \frac{N}{\log(N)} N^{-\alpha} \leq N^{1-\alpha}. \tag{18}$$

Substituting $\alpha = (1 - \log_N(\epsilon))$ yields

$$\mathbb{P}\left[\rho(X_N) > 2\Phi^{-1}\left(\frac{\log(N\epsilon^{-1})}{N}\right)\right] \leq \epsilon, \tag{19}$$

from which the general claim follows.

The result is specialised to the unit hypercube $[0,1]^d$ simply by observing that in this case, one can take

$$\Phi(r) := \frac{\pi^{d/2}}{2^d \Gamma(d/2 + 1)} r^d, \tag{20}$$

which is the volume of the intersection of the ball of radius r, centered at a corner, with the cube.

For the unit hypersphere $S^d \subset \mathbb{R}^{d+1}$, fix $x \in S^d$ and consider the Euclidean ball $B(x,r) \subset \mathbb{R}^{d+1}$ centered at x. It is clear that the intersection $B_r := S^d \cap B(x,r)$ is a geodesic for the inherited Riemannian metric on S^d. In particular, for $r < 2$, B_r is a hyperspherical cap. Elementary trigonometry shows that the angle subtended by the line connecting x to 0 and the line connecting any point on the boundary of B_r to 0 is given by $2\sin^{-1}(r/2)$. It then follows from (Li, 2011, p. 67) and the elementary trigonometric identity $\sin(2\theta) = 2\sin(\theta)\cos(\theta)$ that one has

$$\mathbb{P}[B(x,r)] = C^{-1} \frac{\pi^{\frac{d+1}{2}}}{\Gamma\big(\frac{d+1}{2}\big)} I_{r^2 - r^4/4}\left(\frac{d}{2}, \frac{1}{2}\right),$$

where

$$I_x(a,b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \int_0^x t^{a-1}(1-t)^{b-1}\,dt$$

is the regularised incomplete beta function and $C = 2\pi^{\frac{d}{2}}\Gamma(d/2)^{-1}$ is the volume of the unit d-sphere. For any $0 < t < 1$ one has $(1-t)^{1-\frac{1}{2}} \geq 1$, thus one has the estimate

$$I_{r^2 - r^4/4}\left(\frac{d}{2}, \frac{1}{2}\right) \geq \frac{\Gamma\big(\frac{d+1}{2}\big)}{\Gamma\big(\frac{d}{2}\big)\Gamma\big(\frac{1}{2}\big)} (r^2 - r^4/4)^{\frac{d}{2}}.$$

Now, fixing $0 < \gamma < 1$ and assuming $r^2/4 < \gamma$, we therefore have

$$\mathbb{P}(B(x,r)) \geq \frac{1}{2}(1-\gamma)^{\frac{d}{2}} r^d.$$

We can then use $\Phi(r) = \frac{1}{2}(1-\gamma)^{\frac{d}{2}} r^d$ in defining $\delta(N,\epsilon) = 2\Phi^{-1}(\log(N\epsilon^{-1})N^{-1})$ as in Equation (19) provided only that the assumption $\delta^2/4 < \gamma$ is satisfied. Setting $\delta = 2^{1+\frac{1}{d}}(1-\gamma)^{-\frac{1}{2}}(\log(N\epsilon^{-1})N^{-1})^{\frac{1}{d}}$, this requirement reduces to having N sufficiently large that $2^{\frac{2}{d}}(\log(N\epsilon^{-1})N^{-1})^{\frac{2}{d}} < \gamma - \gamma^2$.

For the final result, when pushing forward a good distribution by a Lipschitz function, the stated result follows from the Lipschitz identity. ☐

Proof of Theorem 4.3. Fix $N \in \mathbb{N}$ and $0 < \epsilon < 1$. By δ-goodness of the data distribution, with probability at least $1-\epsilon$ over i.i.d. samples $(x_1, \dots, x_N)$, the balls $\{B(x_i, \delta(N,\epsilon))\}_{i=1}^N$ cover $\mathrm{supp}(\mathbb{P})$. For notational convenience, denote $B(x_i, \delta(N,\epsilon))$ by simply B_i. Conditioning on the event that the B_i cover $\mathrm{supp}(\mathbb{P})$, one has

$$\begin{aligned}
\|f\|_{Lip,\mathbb{P}} &\leq \sup_{x \in \bigcup_{i=1}^N B_i} \|Jf(x)\|_2 = \max_i \sup_{x \in B_i} \|Jf(x)\|_2 \\
&\leq \max_i \sup_{x \in B_i} \|Jf(x_i) - Jf(x_i) + Jf(x)\|_2 \\
&\leq \max_i \left(\|Jf(x_i)\|_2 + \sup_{x \in B_i} \|Jf(x) - Jf(x_i)\|_2 \right) \\
&\leq \max_i \big(\|Jf(x_i)\|_2 + V_{B_i}(Jf) \big)
\end{aligned}$$

The result follows. ☐

Proof of Theorem 5.1. Recall that by hypothesis on c, one has $c(z_1, z_2) \geq \alpha \|z_1 - z_2\|_2^2$ for all z_1, z_2 in the domain of c. Thus, since $c(f(x_i), f^*(x_i)) < \ell^2 \alpha$ for all i, we therefore have

$$\|f(x_i) - f^*(x_i)\|_2 \leq \ell \tag{21}$$

for all i. That is, for each i, $f(x_i)$ is contained in the ℓ-ball centred on the target value $f^*(x_i)$.

Since the smallest distance between any two $f(x_i)$ and $f(x_j)$ is realised when these points lie on the straight line between $f^*(x_i)$ and $f^*(x_j)$, one has the lower bound

$$\|f(x_i) - f(x_j)\|_2 \geq \|f^*(x_i) - f^*(x_j)\|_2 - 2\ell \tag{22}$$

for all i, j. Invoking the Lipschitz identity, and using the fact that $x_i = x_j$ for $i \neq j$ with probability zero, then gives

$$\|f\|_{Lip,\mathbb{P}} \geq \max_{i \neq j} \frac{\|f^*(x_i) - f^*(x_j)\|_2 - 2\ell}{\|x_i - x_j\|_2}. \tag{23}$$

Conditioning now on the event that $\mathrm{supp}(\mathbb{P}) \subset \bigcup_{i=1}^N B(x_i, \delta(\epsilon, N))$, which occurs with probability at least $1-\epsilon$ by δ-goodness of $\mathbb{P}$, and invoking Theorem 4.3 together with the subadditivity of the component-wise maximum function, gives the result. ☐

Proof of Theorem 6.1. Fix $N \in \mathbb{N}$ and $0 < \epsilon < 1$. Since $\mathbb{P}$ is δ-good, with probability at least $1-\epsilon$ over i.i.d. samples $(x_1, \dots, x_N)$, for each $x \in \mathrm{supp}(\mathbb{P})$ there exists j such that $\|x - x_j\|_2 \leq \delta(N,\epsilon)$. Conditioning on this event and fixing $x \in \mathrm{supp}(\mathbb{P})$ with corresponding nearby x_j, the triangle inequality applies to yield

$$\|f(x) - f^*(x)\|_2 \leq \|f(x) - f(x_j)\|_2 + \|f(x_j) - f^*(x_j)\|_2 + \|f^*(x_j) - f^*(x)\|_2, \tag{24}$$

which gives

$$\|f(x) - f^*(x)\|_2 \leq \delta(N,\epsilon)\big(\|f\|_{Lip,\mathbb{P}} + \|f^*\|_{Lip}\big) \tag{25}$$

after applying the Lipschitz identities and invoking the hypothesis that f and f^* agree on training data. The proof of Theorem 4.3 now applies to show that, conditioned on this same event, one has

$$\|f(x) - f^*(x)\|_2 \leq \delta(N,\epsilon)\big(\|f^*\|_{Lip} + \max_i \big(\|Jf(x_i)\|_2 + V_{B_i}(Jf)\big)\big). \tag{26}$$

Taking the $\mathbb{P}$-expectation of both sides yields the result. ☐

A.1 A NOTE ON JACOBIAN COMPUTATIONS

In our experiments, the Jacobian norm and sharpnesss are estimated using the power method and the Hessian/Jacobian-vector product functions in PyTorch. Recalling that we use $\mathbb{R}^{d\times N}$ to denote the set of data matrices, consisting of a batch of data column vectors concatenated together, there are two different kinds of functions $f : \mathbb{R}^{d_0\times N} \to \mathbb{R}^{d_1\times N}$ whose Jacobians we would like to compute. The first of these are functions defined columnwise by some other map $g : \mathbb{R}^{d_0} \to \mathbb{R}^{d_1}$: that is

$$f(X)_j = g(X_j), \tag{27}$$

where lower indices index columns so that $X_j \in \mathbb{R}^{d_0}$ for all j. In this case, flattening (columns first) shows that $Jf(X)$ is just a block-diagonal matrix, whose j^{th} block is $Jf(X_j)$. Hence $\|Jf(X)\|_2 = \max_j \|Jf(X_j)\|_2$. For a network in evaluation mode, *every* layer Jacobian is of this form.

On the other hand, for a network in *train* mode (as is the case when evaluating the Hessian of the loss), batch normalisation (BN) layers do not have this form. BN layers in train mode are *nonlinear* transformations defined by

$$bn(X) := \frac{X - \mathbb{E}X}{(\epsilon + \sigma^2 X)^{\frac{1}{2}}}, \tag{28}$$

where $\mathbb{E}$ and σ^2 denote mean and variance computed across the column index (again we omit the parameters as they are not essential to the discussion). The skeptical reader would rightly question whether the Jacobian of BN in train mode (to which Equation (4) applies) is sufficiently close to the Jacobian of BN in evaluation mode (to which Theorem 6.1 applies) for Ansatz 3.1 to be valid for BN networks. Our next and final theorem says that for sufficiently large datasets, this is not a problem.

Theorem A.2. *Let bn_T and bn_V denote a batch normalisation layer in train and evaluation mode respectively. Given a data matrix $X \in \mathbb{R}^{d\times N}$, assume the coordinates of X are $O(1)$. Then $Jbn_T = Jbn_V + O(N^{-1})$.*

Proof of Theorem A.2. Given a data matrix X, let its row-wise mean and variance, thought of as *constant vectors*, be denoted by $\mathbb{E}$ and σ^2. The evaluation mode BN map $bn_V : \mathbb{R}^{d\times N} \to \mathbb{R}^{d\times N}$ is simply an affine transformation, with Jacobian coordinates given by

$$\frac{\partial (bn_V)^i_j}{\partial x^k_l} = \frac{\delta^i_k \delta^i_l}{(\epsilon + \sigma^2)^{\frac{1}{2}}}. \tag{29}$$

By (MacDonald et al., 2022, Lemma 8.3), the train mode BN map $bn_T : \mathbb{R}^{d\times N} \to \mathbb{R}^{d\times N}$ can be thought of as the composite $v \circ m$, where $m, v : \mathbb{R}^{d\times N} \to \mathbb{R}^{d\times N}$ are defined by

$$m(X) := X - \frac{1}{N} X 1_{N\times N} \tag{30}$$

and

$$v(Y) := (\epsilon + N^{-1}\|Y\|^2_{row})^{-\frac{1}{2}} Y \tag{31}$$

respectively. Here $1_{N\times N}$ denotes the $N \times N$ matrix of 1s, and $\|Y\|_{row}$ denotes the vector of row-wise norms of Y.

As in (MacDonald et al., 2022, Lemma 8.3), one has

$$\frac{\partial v^i_j}{\partial y^k_l}(Y) = \frac{\delta^i_k}{(\epsilon + N^{-1}\|Y^i\|^2)^{\frac{1}{2}}} \left(\delta^j_l - \frac{y^i_l y^i_j}{(\epsilon + N^{-1}\|Y^i\|^2)} \right), \qquad \frac{\partial m^i_j}{\partial x^k_l}(Y) = \delta^i_k(\delta^i_l - N^{-1}) \tag{32}$$

for any data matrix Y. Thus, invoking the chain rule and simplifying, we therefore get

$$\frac{\partial (bn_T)^i_j}{\partial x^k_l}(X) = \frac{\delta^i_k \delta^i_l}{(\epsilon + \sigma^2)^{\frac{1}{2}}} - \frac{1}{N} \frac{\delta^i_k (x^i_l - \mathbb{E}^i)(x^i_l - \mathbb{E}^i)}{(\epsilon + \sigma^2)^{\frac{3}{2}}} + O(N^{-1}) \tag{33}$$

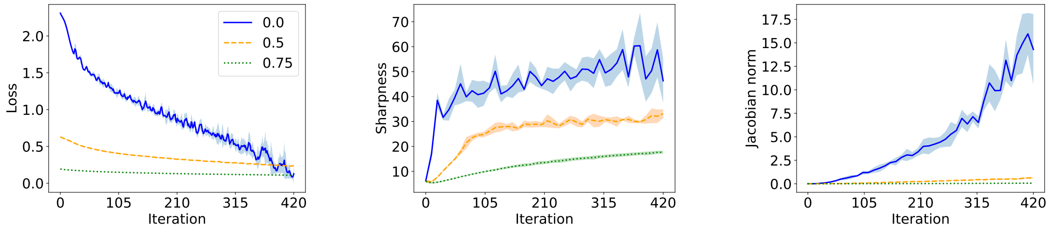
Since the components x are $O(1)$ by hypothesis, the result follows. □

B ADDITIONAL EXPERIMENTAL RESULTS AND DETAILS

B.1 PROGRESSIVE SHARPENING

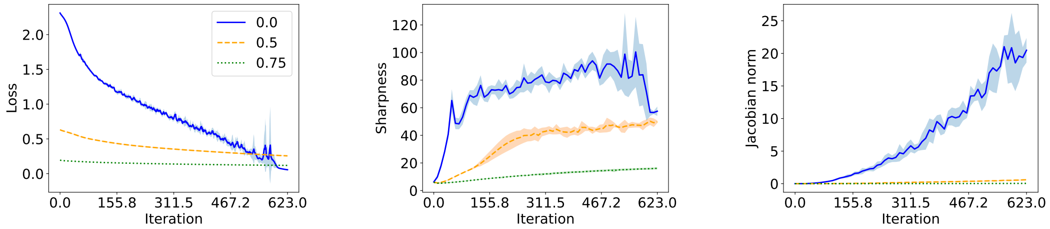
We trained ResNet18 and VGG11 (superficially modifying `https://github.com/kuangliu/pytorch-cifar/blob/master/models/resnet.py` and `https://github.com/chengyangfu/pytorch-vgg-cifar10/blob/master/vgg.py` respectively, with VGG11 in addition having its dropout layers removed, but BN layers retained) with full batch gradient descent on CIFAR10 with learning rates 0.08, 0.04 and 0.02, using label smoothings of 0.0, 0.5 and 0.75. The models with no label smoothing were trained to 99 percent accuracy, and the number of iterations required to do this were then used as the number of iterations to train the models using nonzero label smoothing. Sharpness and Jacobian norm were computed every 5, 10, or 20 iterations depending on the number of iterations required for convergence of the non-label-smoothed model. Five trials were conducted in each case, with mean and standard deviation plotted. Line style indicates degree of label smoothing.

The dataset was standardised so that each vector component is centered, with unit standard deviation. Full batch gradient descent was approximated by averaging the gradients over 10 “ghost batches” of size 5000 each, as in Cohen et al. (2021). This is justified with BN networks by Theorem A.2. The Jacobian norm we plot is for the Jacobian of the softmaxed model, as required by our derivation of Equation (9). The sharpness and Jacobian norms were estimated using a randomly chosen subset of 2500 data points. In all cases, the smoother labels are associated with less severe increase in Jacobian and sharpness during training, as predicted by Equation (9). Refer to `fullbatch.py` in the supplementary material for the code we used to run these experiments.



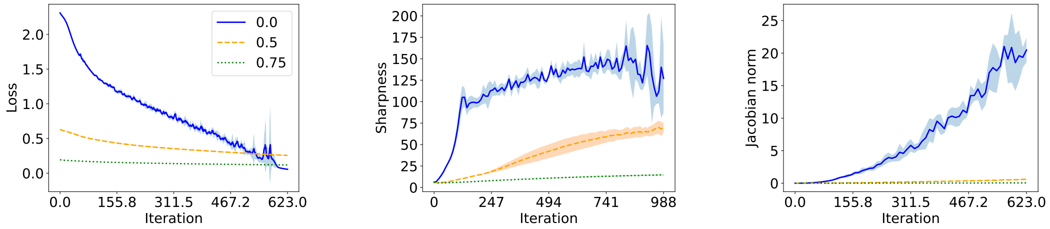
(a) Loss (b) Sharpness (c) Jacobian norm

Figure 6: VGG11, learning rate 0.08



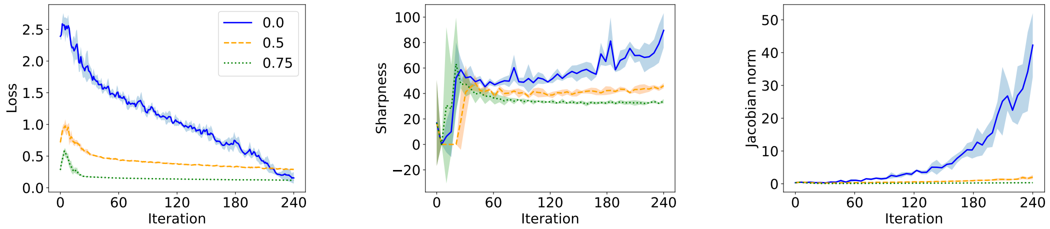
(a) Loss (b) Sharpness (c) Jacobian norm

Figure 7: VGG11, learning rate 0.04



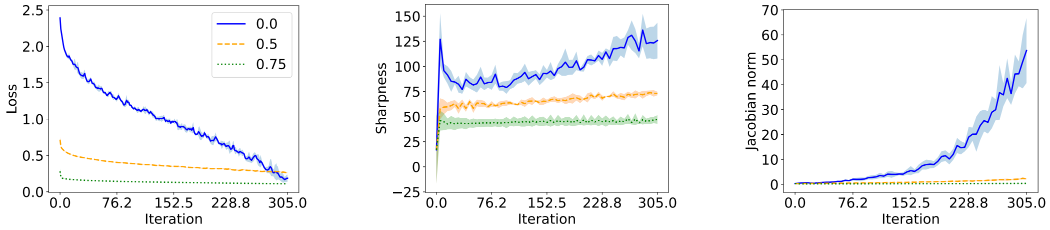
(a) Loss (b) Sharpness (c) Jacobian norm

Figure 8: VGG11, learning rate 0.02



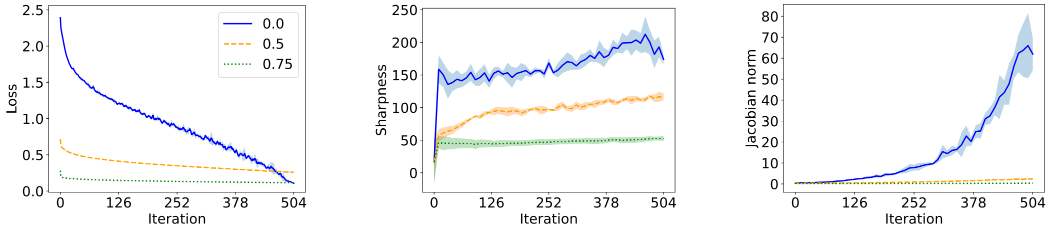
(a) Loss (b) Sharpness (c) Jacobian norm

Figure 9: ResNet18, learning rate 0.08



(a) Loss (b) Sharpness (c) Jacobian norm

Figure 10: ResNet18, learning rate 0.04



(a) Loss (b) Sharpness (c) Jacobian norm

Figure 11: ResNet18, learning rate 0.02

B.2 BATCH SIZE AND LEARNING RATE

We trained a VGG11 and ResNet18 (superficially modifying `https://github.com/chengyangfu/pytorch-vgg-cifar10/blob/master/vgg.py` and `https://github.com/kuangliu/pytorch-cifar/blob/master/models/resnet.py` respectively, additionally removing dropout and retaining BN in the VGG11) on CIFAR10 and CIFAR100 at constant learning rates 0.1, 0.01, and 0.001, with batch sizes of 64, 128, 256, 512 and 1024. The data were normalised by their RGB-channelwise mean and standard deviation, computed across all pixel coordinates and all elements of the training set. Five trials of each were conducted. Training loss on the full data set was computed every 100 iterations, and training was terminated when the average of the most recent 10 such losses was smaller than 0.01 for CIFAR10 and 0.02 for CIFAR100. Weight decay regularisation of 0.0001 was used throughout. Refer to `batch_lr.py` in the supplementary material.

At the termination of training, training loss and test loss were computed, with estimates of the Jacobian norm and sharpness computed using a randomly selected subset of size 2000 from the training set. The same seed was used to generate the parameters for a given trail as was used to generate the random subset of the training set.

We see that, on the whole, the claim that smaller batch sizes and larger learning rates lead to flatter minima and better generalisation is supported by our experiments, with Jacobian norm in particular being well-correlated to generalisation gap as anticipated by Theorem 6.1 and Ansatz 3.1. As noted in the main body of the paper, the notable exception to this is ResNet models trained with the largest learning rate, where Jacobian norm appears to underestimate the Lipschitz constant of the model and hence the generalisation gap. We speculate that this is due to large learning rate training of skip connected models leading to larger Jacobian Lipschitz constant, making the Jacobian norm a poorer estimate of the Lipschitz constant of the model and hence of generalisation (Theorems 4.3 and 6.1).

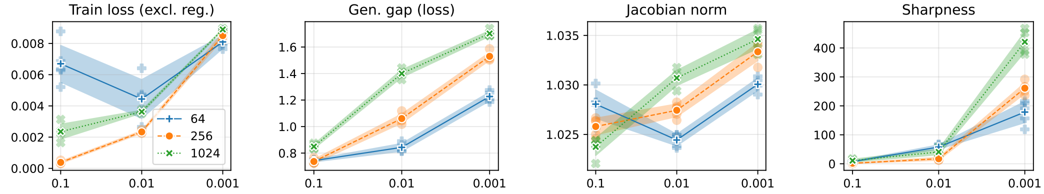


Figure 12: ResNet18 on CIFAR10, learning rate on x axis. Line style indicates batch size.

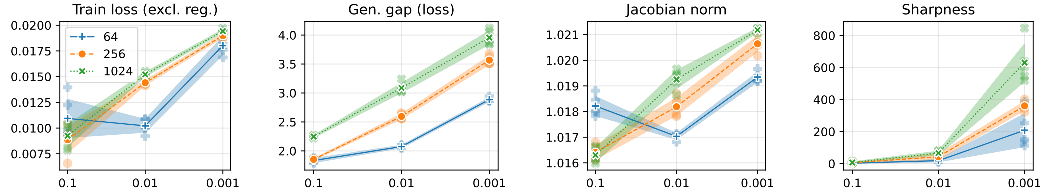


Figure 13: ResNet18 on CIFAR100, learning rate on x axis. Line style indicates batch size.

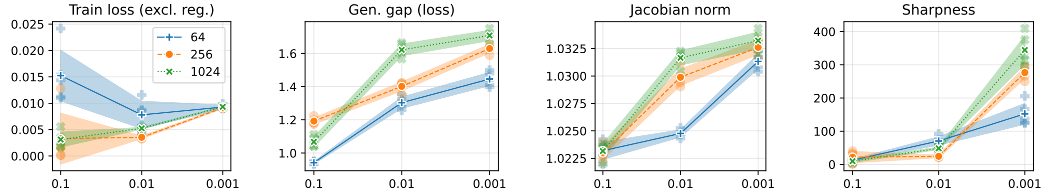


Figure 14: VGG11 on CIFAR10, learning rate on x axis. Line style indicates batch size.

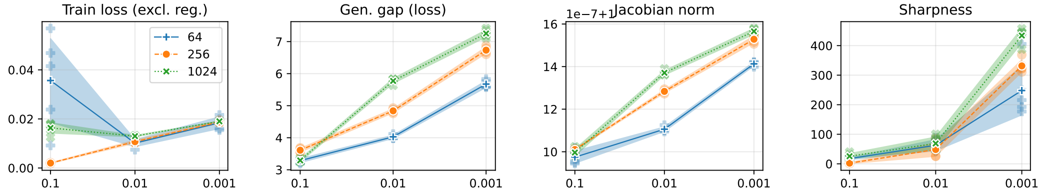


Figure 15: VGG11 on CIFAR100, learning rate on x axis. Line style indicates batch size.

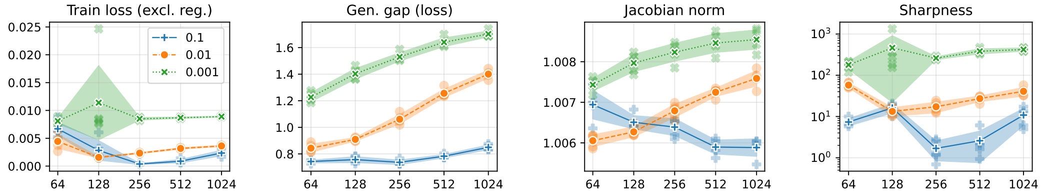


Figure 16: ResNet18 on CIFAR10, batch size on x axis. Line style indicates learning rate.

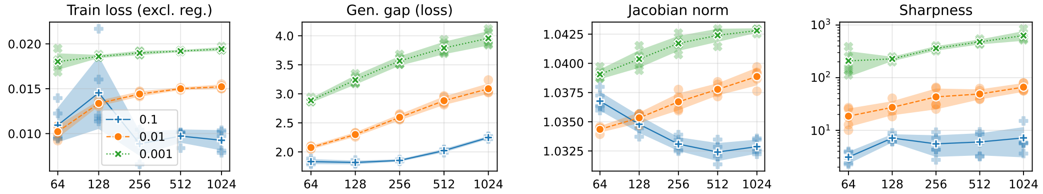


Figure 17: ResNet18 on CIFAR100, batch size on x axis. Line style indicates learning rate.

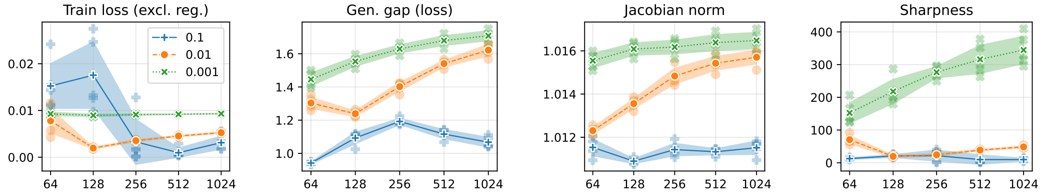


Figure 18: VGG11 on CIFAR10, batch size on x axis. Line style indicates learning rate.

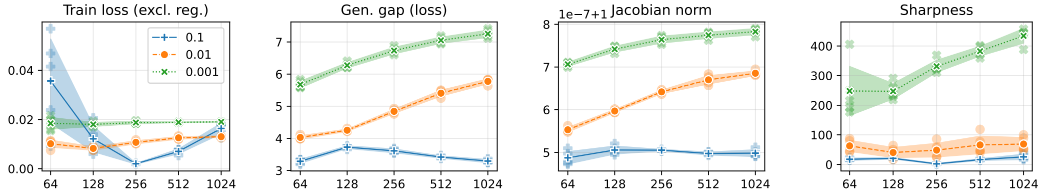


Figure 19: VGG11 on CIFAR100, batch size on x axis. Line style indicates learning rate.

B.3 EXTENDED PRACTICAL RESULTS

Here we provide the experimental details and additional experimental results for Section 6. In addition to the generalisation gap, sharpness, and (input-output) Jacobian norm, these plots include the operator norm of the Gauss-Newton matrix, the loss on the training set, and the accuracy on the validation set. Note that the figures in this section present the *final* metrics at the conclusion of training as a function of the regularisation parameter, and hence progressive sharpening (as a function of time) will not be observed directly. Rather, the purpose is to examine the impact of different regularisation strategies on the Hessian and model Jacobian.

In all cases, the model Jacobian is correlated with the generalisation gap, whereas the Hessian is often uncorrelated. Moreover, the norm of the Gauss-Newton matrix is (almost) always very similar to that of Hessian, in line with previous empirical observation Papyan (2018; 2019); Cohen et al. (2021). The results support the arguments that (i) the model Jacobian is strongly related to the generalisation gap and (ii) while the Hessian is connected to the model Jacobian, and forcing the Hessian to be sufficiently small likewise appears to force the Jacobian to be smaller (see the SAM plots), the Hessian is also influenced by other factors and can be large even without the Jacobian being large (see other plots). All of these phenomena are consistent with our ansatz.

The figures in this section present results for $\{$CIFAR10, CIFAR100$\}$ $\times$ $\{$VGG11, ResNet18$\}$ $\times$ $\{$batch-norm, no batch-norm$\}$. Line colour and style denote different initial learning rates. The models were trained for 90 epochs with a minibatch size of 128 examples using Polyak momentum of 0.9. The learning rate was decayed by a factor of 10 after 50 and 80 epochs. Models were trained using the softmax cross-entropy loss with weight decay of 0.0005. (If weight decay were disabled, we would often observe the Hessian to collapse to zero as the training loss went to zero, due to the vanishing second gradient of the cost function in this region (Cohen et al., 2021, Appendix C). This would make the correlation between sharpness and generalisation gap even worse.) Refer to `train_jax.py` and `slurm/launch_wd_sweep.sh` for the default configuration and experiment configuration, respectively.

For each regularisation strategy, the degree of regularisation is parametrised as follows. Label smoothing considers smoothed labels $(1-\alpha)y + \alpha(1/n)$ with $\alpha \in [0, 1]$. Mixup takes a convex combination of two examples $(1-\theta)(x, y) + \theta(x', y')$. The coefficient θ is drawn from a symmetric beta distribution $\theta \sim \mathrm{Beta}(\beta, \beta)$ with $\beta \in [0, \infty)$, which corresponds to a Bernoulli distribution when $\beta = 0$ and a uniform distribution when $\beta = 1$. Data augmentation modifies the inputs x with probability $p \in [0, 1]$; the image transforms which we adopt are the standard choices for CIFAR (four-pixel padding, random crop, random horizontal flip). Sharpness Aware Minimisation (SAM) restricts the distance ρ between the current parameter vector and that which is used to compute the update, which simplifies to SGD when $\rho = 0$.

For all results presented in this section, the loss and generalisation gap were computed using “clean” training examples; that is, without mixup or data augmentation in the respective experiments (despite this requiring an additional evaluation of the model for each step). The Hessian, Gauss-Newton and input-output Jacobian norms were similarly computed using a random batch of 1000 clean training examples. The losses and matrix norms in this section similarly exclude weight decay. Label smoothing was interpreted as a modification of the loss function rather than the training distribution, and therefore *was* included in the calculation of the losses and matrix norms. Batch-norm layers were used in “train mode” (statistical moments computed from batch) for the Hessian and Gauss-Newton matrix, and in “eval mode” (statistical moments are constants) for the Jacobian matrix, although this difference was observed to have negligible effect on the Jacobian norm.

B.3.1 CIFAR10

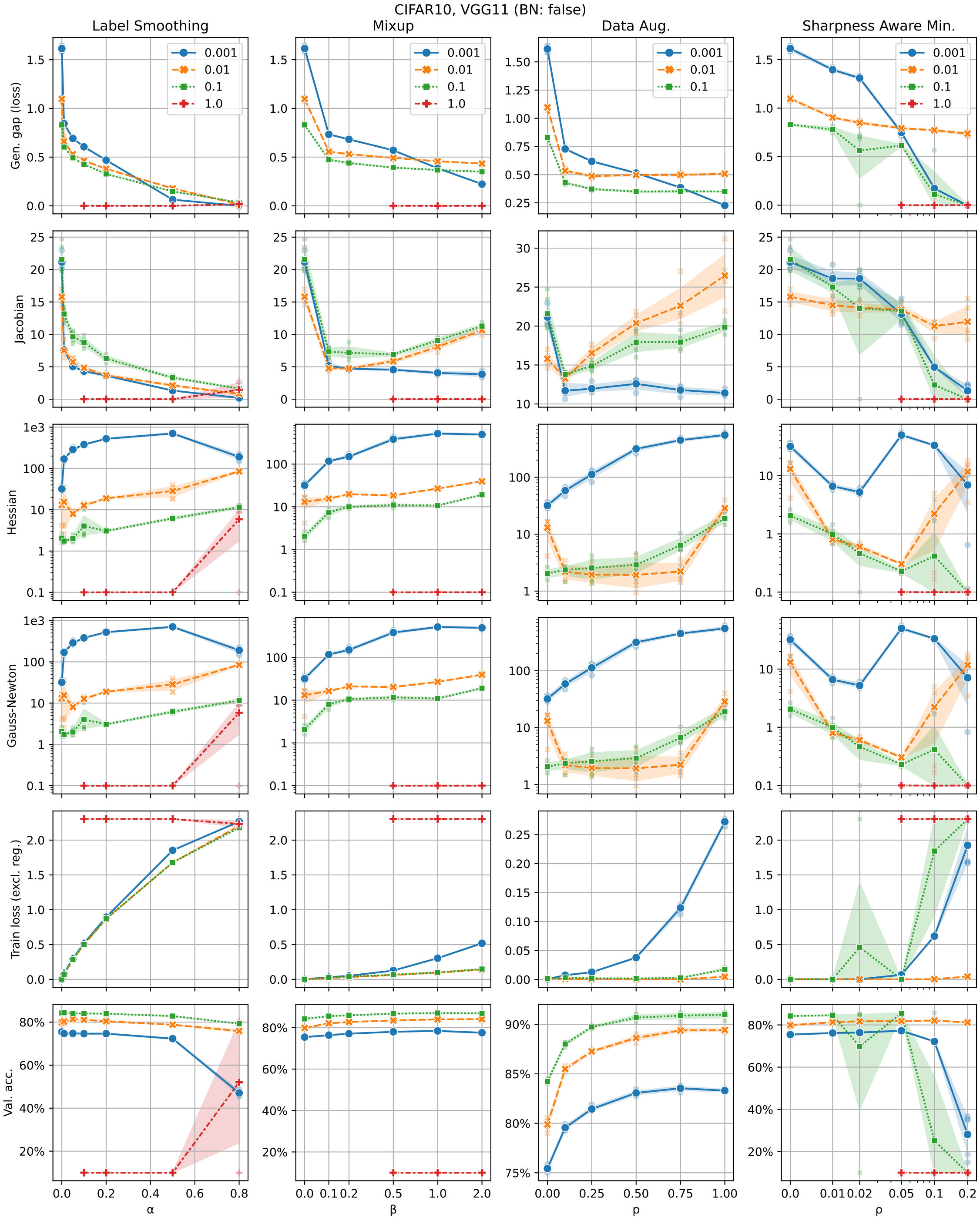


Figure 20: VGG11 without batch-norm on CIFAR10.

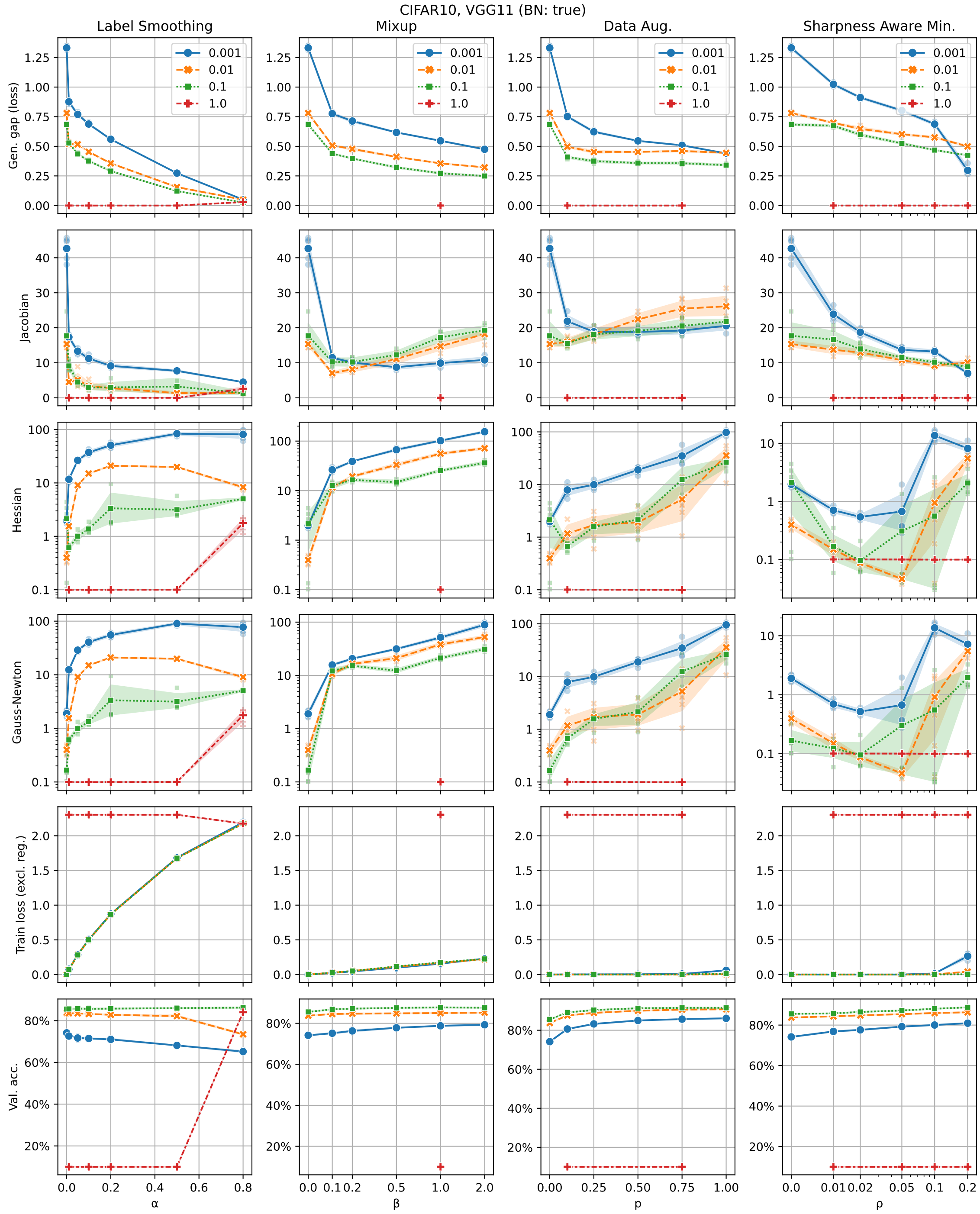


Figure 21: VGG11 with batch-norm on CIFAR10.

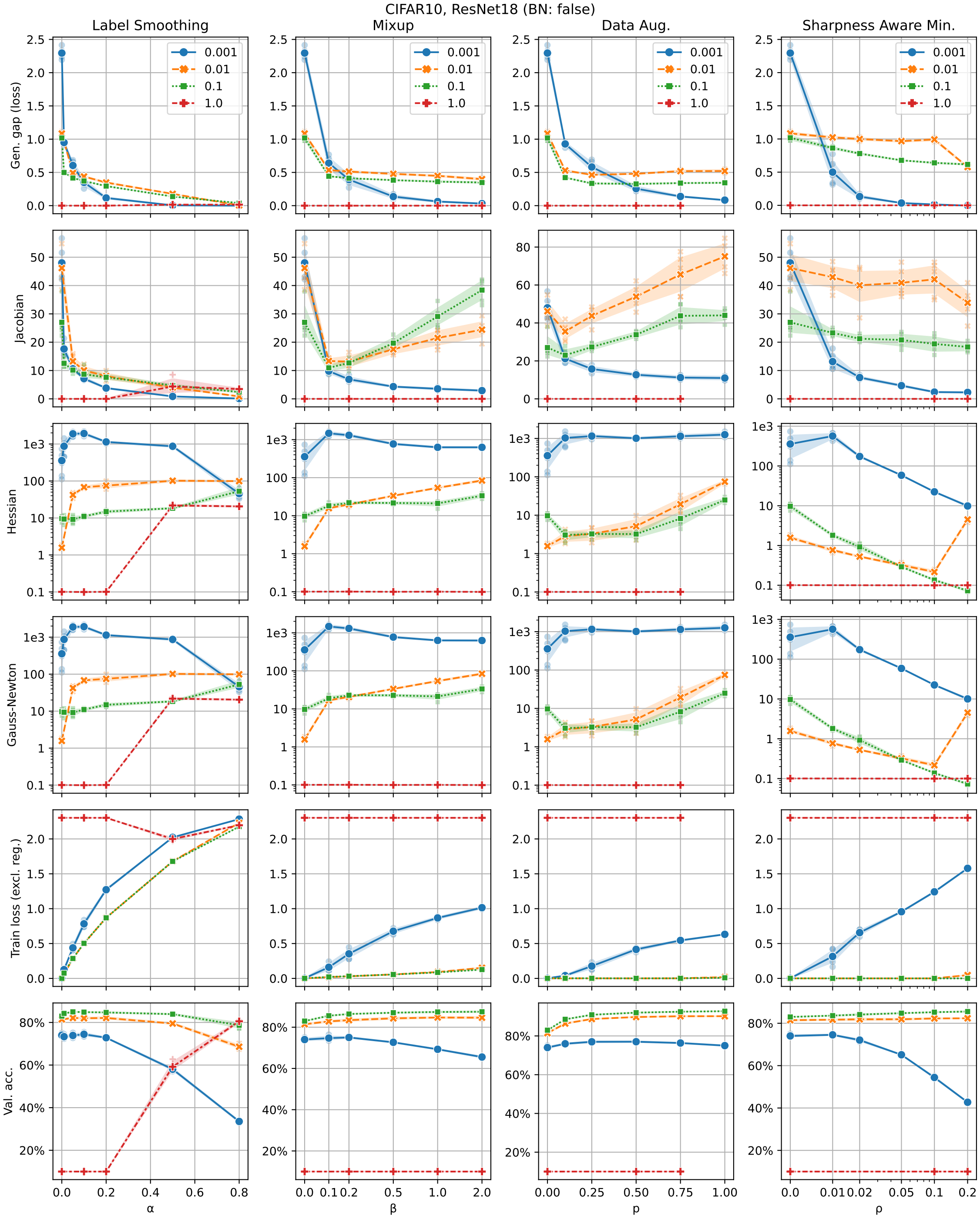


Figure 22: ResNet18 without batch-norm on CIFAR10.

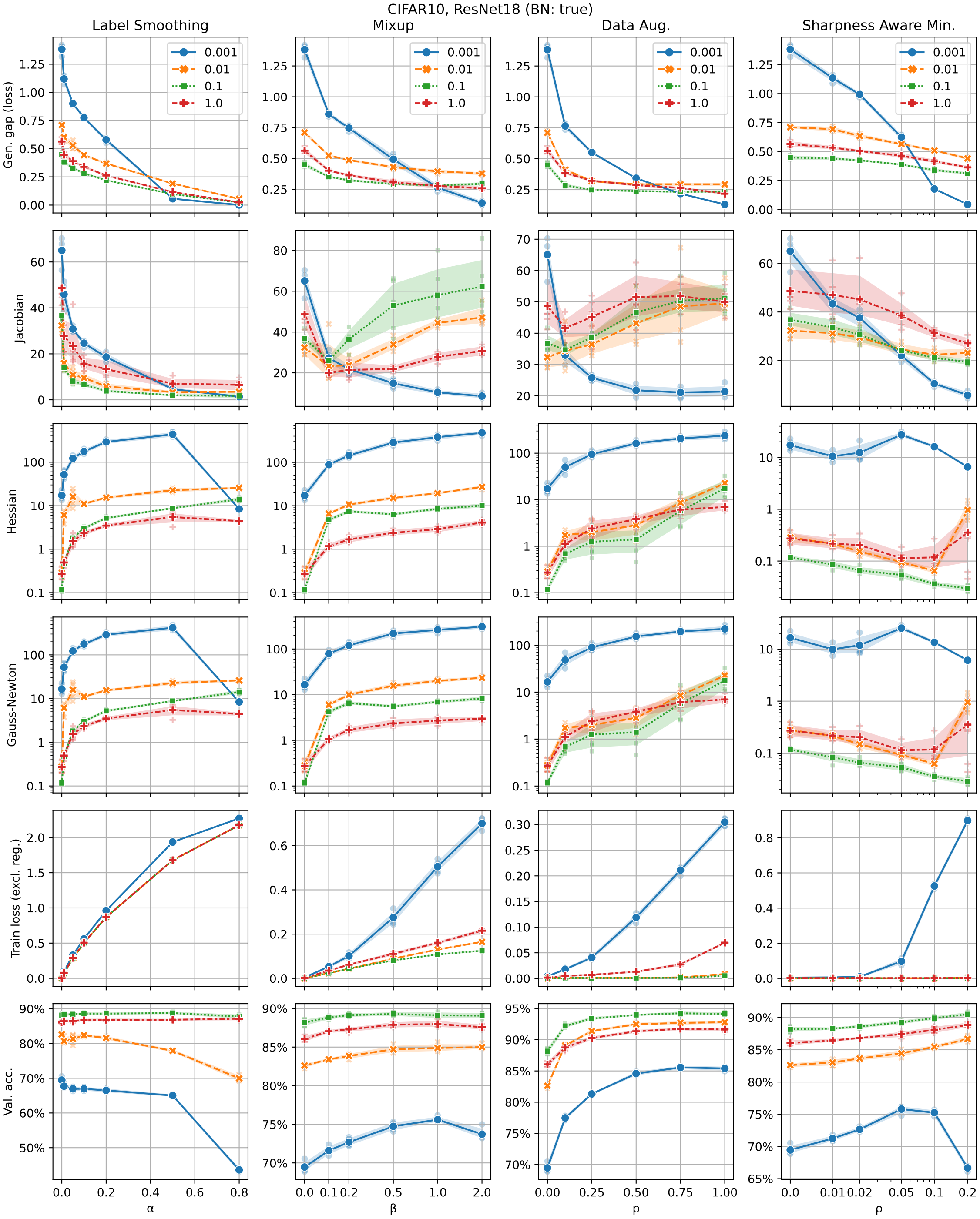


Figure 23: ResNet18 with batch-norm on CIFAR10.

B.3.2 CIFAR100

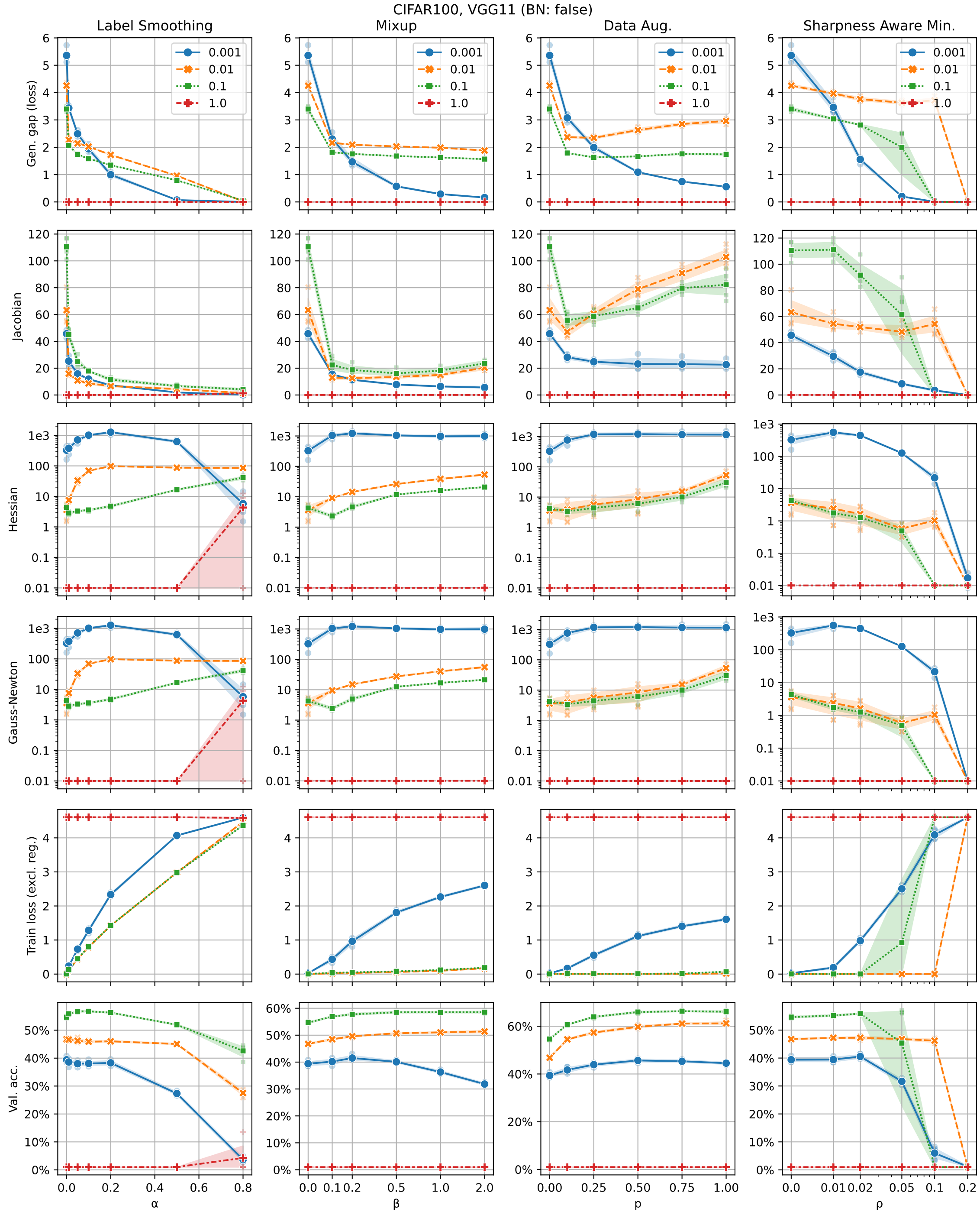


Figure 24: VGG11 without batch-norm on CIFAR100.

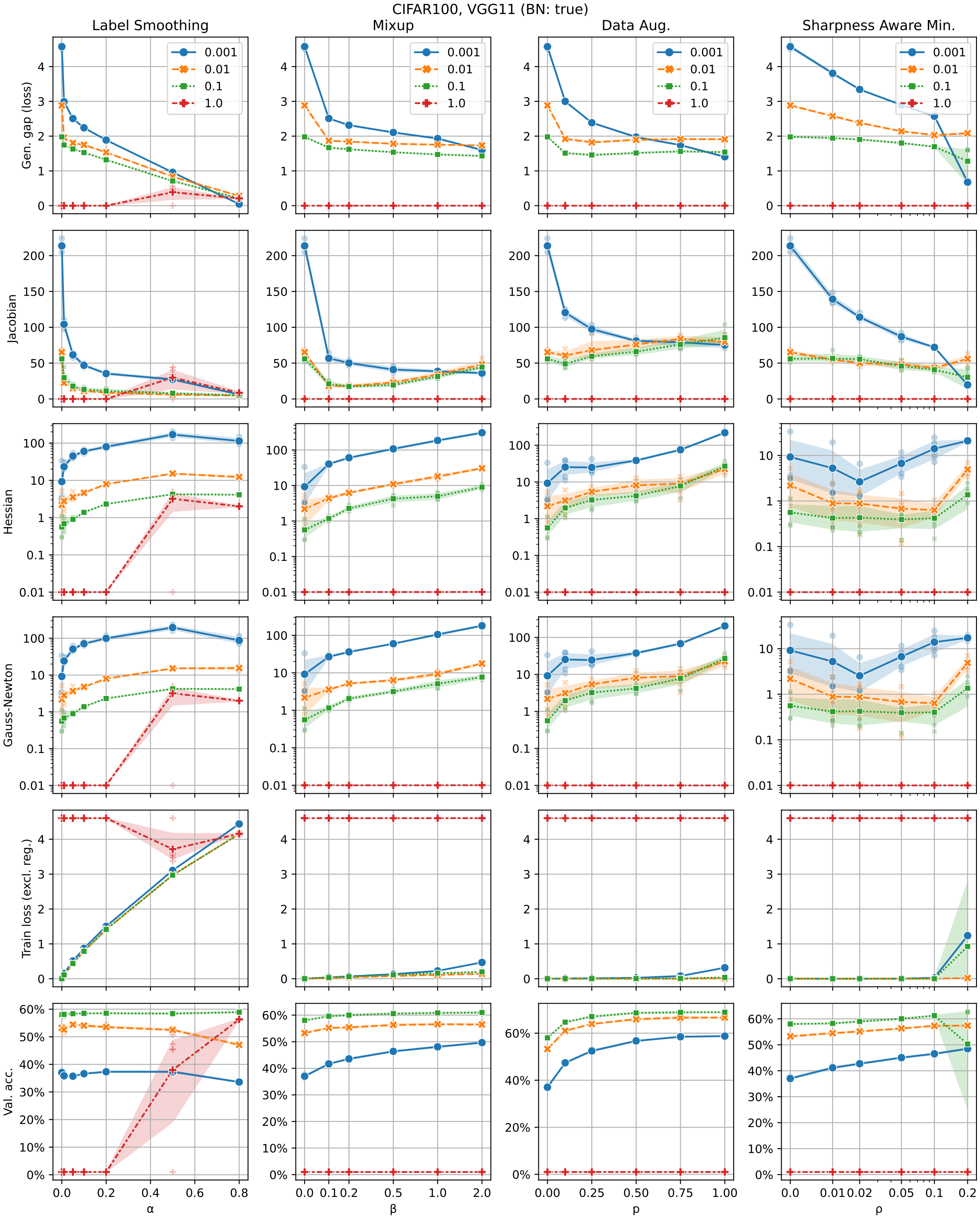


Figure 25: VGG11 with batch-norm on CIFAR100.

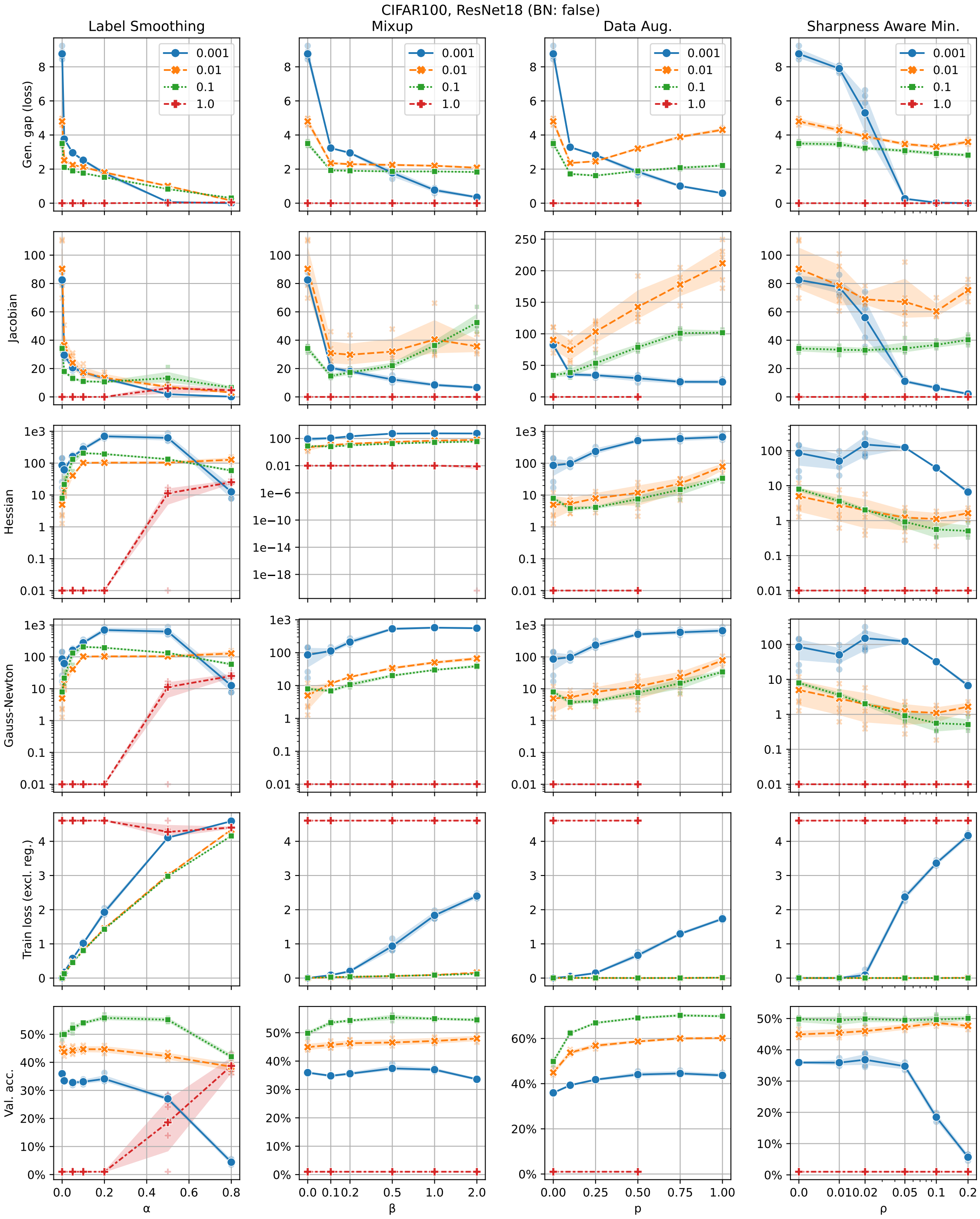


Figure 26: ResNet18 without batch-norm on CIFAR100.

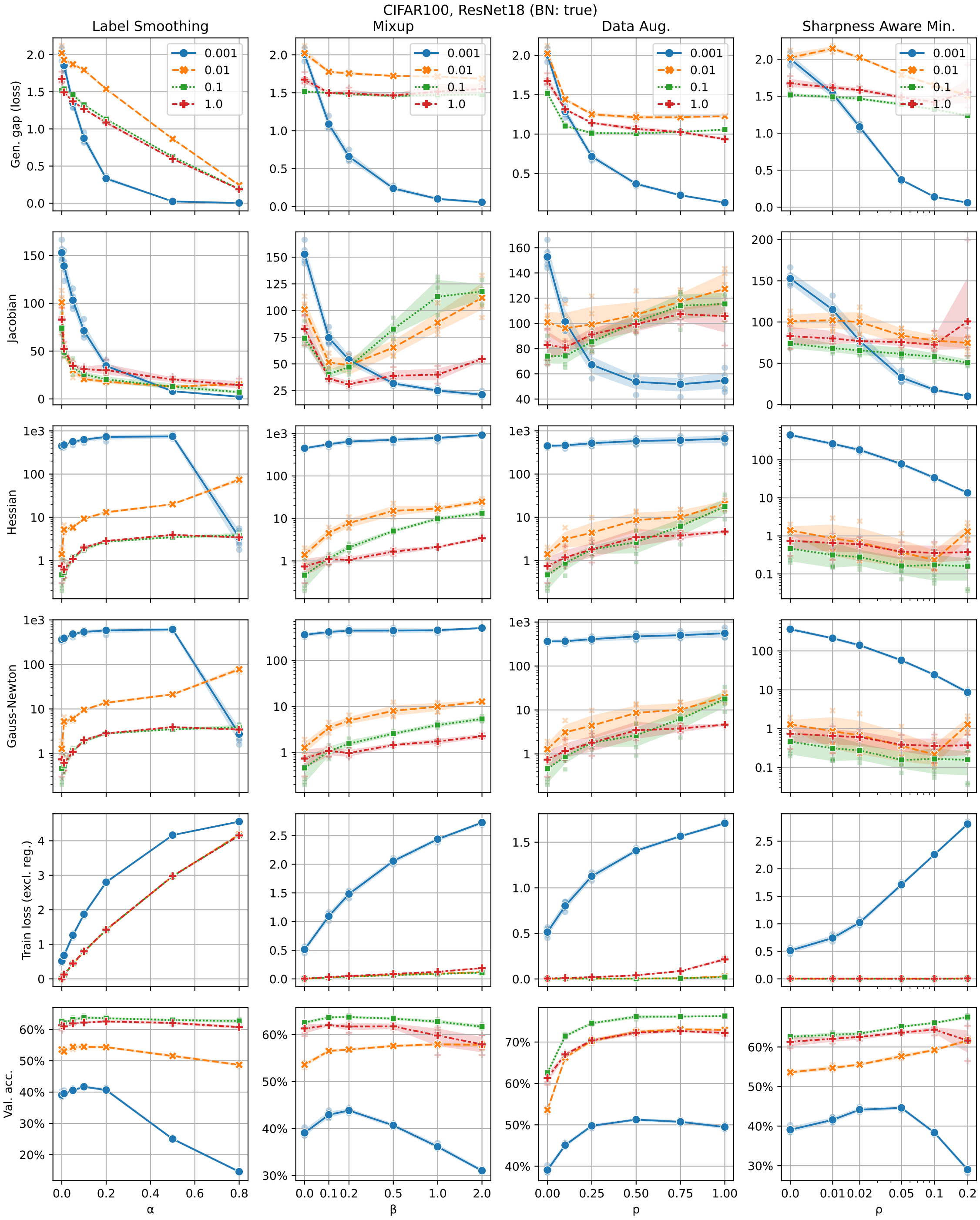


Figure 27: ResNet18 with batch-norm on CIFAR100.

C ON MEDIATING FACTORS IN THE ANSATZ

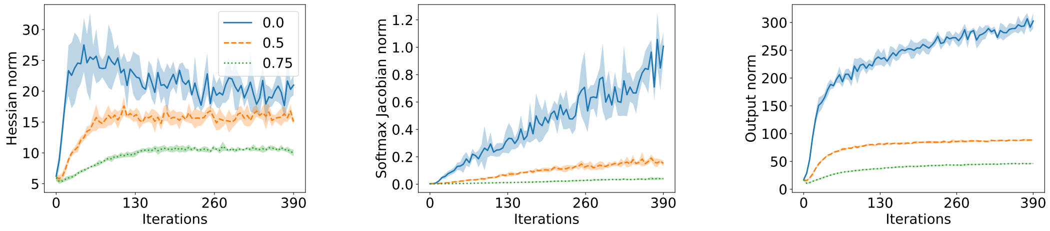
In this section we provide classification and regression experiments to show that the relationship between loss curvature and input-output Jacobians is not always simple. Recall again Equation (4):

$$C\, DF_X\, DF_X^T\, C^T = C\left(\sum_{l=1}^{L}\left(Jf_L \cdots Jf_{l+1}\, Df_l\, Df_l^T\, Jf_{l+1}^T \cdots Jf_L^T\right)\right)C^T. \tag{34}$$

Due to the presence of the parameter derivatives Df_l and the square root C of the Hessian of the cost function, as well as the absence of the first layer Jacobian in Equation (4), it is not always true that the magnitude of the Hessian and that of the model's input-output Jacobian are correlated.

C.1 THE COST FUNCTION

We trained a VGG11, again using `https://github.com/chengyangfu/pytorch-vgg-cifar10/blob/master/vgg.py` with droupout layers removed and BN layers retained, on CIFAR10 using SGD with a batch size of 128, using differing degrees of label smoothing. Data was standardised so that each RGB channel has zero mean and unit standard deviation over all pixel coordinates and training samples. In Figure 28 we plot the sharpness and Jacobian norm every five iterations in the first epoch, observing exactly the same relationship between label smoothing, Jacobian norm and sharpness as predicted in Section 5. Refer to `vgg_sgd.py` in the supplementary material for the code to run these experiments.



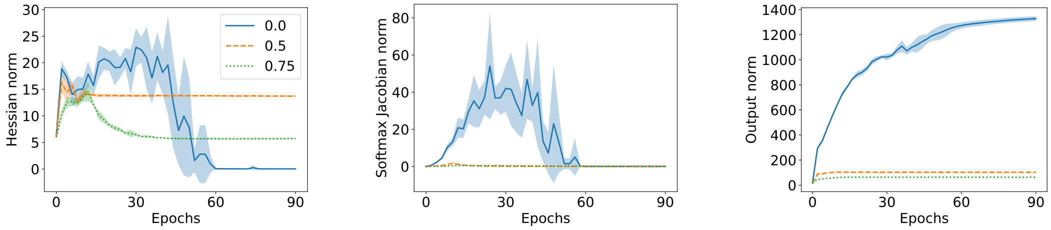
(a) Sharpness

(b) Jacobian norm

(c) Output norm

Figure 28: Effect of label smoothing on sharpness, Jacobian norm and output Frobenius norm during the first epoch of SGD. Line style indicates label smoothing. Exactly as in the full batch GD case, the smoother labels are associated with less severe increase of the Jacobian and less severe progressive sharpening.

Zooming out, however, to the full training run over 90 epochs, Figure 29 shows that the Hessian ultimately behaves markedly differently.



(a) Sharpness

(b) Jacobian norm

(c) Output norm

Figure 29: Effect of label smoothing on sharpness, Jacobian norm and output Frobenius norm during 90 epochs of SGD. Line style indicates label smoothing. When the output norm gets too large, the decay of the C terms in Equation (4) begins to overtake growth in Jacobian norm, ultimately causing the sharpness with unsmoothed labels to collapse to zero, where as sharpness values for smoothed labels plateau at nonzero values. Note also the the Jacobian norm presented is the Jacobian norm of the softmaxed model: the growth in output norm therefore also drives the Jacobian norm to zero.

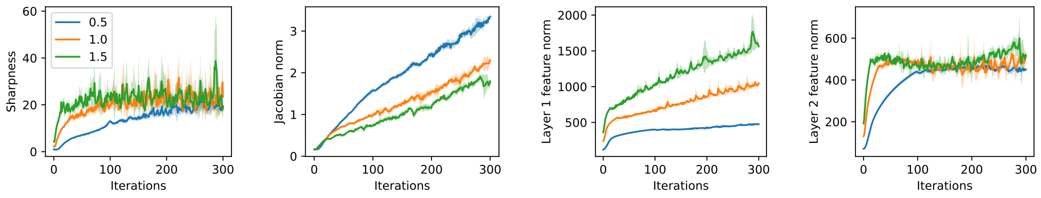
The reason for this is that the norm of the network output grows to infinity in the unsmoothed case (Figure 29), and this growth in output corresponds to a vanishing of the C term (Cohen et al., 2021, Appendix C) corresponding to the cross-entropy cost in Equation (4). Note that this growth is also to blame for the eventual collapse of the norm of the Jacobian of the softmaxed model, since the derivative of softmax vanishes at infinity also. This vanishing of the Jacobian should not be interpreted as saying that the *Lipschitz* constant of the model has collapsed (since if this were the case the model would not be able to fit the data), but rather that the maximum Jacobian norm over training samples has ceased to be a good approximation of the Lipschitz constant of the model, due to the Jacobian itself having a large Lipschitz constant (Theorem 4.3).

C.2 THE PARAMETER DERIVATIVES

Recall Theorem 5.1. We have already shown that decreasing the distance between labels decreases the severity of Jacobian growth, and, in line with Ansatz 3.1, also reduces the severity of progressive sharpening. It seems that one could also manipulate this severity by increasing the distance between input data points, for instance by scaling all data by a constant. We test this training simple, fully-connected, three layer networks, of width 200, with gradient descent on the first 5000 data points of CIFAR10. The data is standardised to have componentwise zero mean and unit standard deviation (measured across the whole dataset). The networks are initialised using the uniform distribution on the interval with endpoints $\pm 1/\sqrt{in\ features}$ (default PyTorch initialisation) and trained with a learning rate of 0.2 for 300 iterations using the cross-entropy cost. Both ReLU and tanh activations are considered. Refer to `small_network.py` in the supplementary material for the code we used to run these experiments.

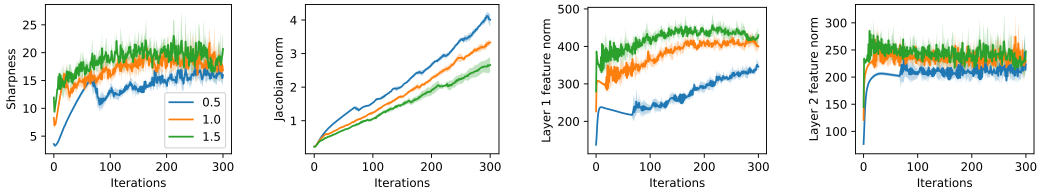
We scale the inputs by factors of 0.5, 1.0 and 1.5, bringing the data closer together at 0.5 and further apart at 1.5. From Theorem 5.1, we anticipate the Jacobian growth to go from most to least severe on these scalings respectively, with sharpness growth behaving similarly according to Ansatz 3.1. Surprisingly, we find that while Jacobian growth *is* more severe for the smaller scalings, the sharpness growth is *less*.

The reason for this is the effect of data scaling on the parameter derivatives Df_l in Equation (4). It is easily computed (cf. MacDonald et al. (2022)) that the parameter derivative Df_l is simply determined by the matrix $f_{l-1}(X)$ of features from the previous layer. In the experimental settings we examined, these matrices are *larger* for the larger scaling values, which pushes the sharpness upwards despite Jacobian norm being smaller. Figures 30 and 31 show this phenomenon on ReLU and tanh activated networks respectively.



(a) Sharpness (b) Jacobian norm (c) Layer 1 feature norm (d) Layer 2 feature norm

Figure 30: Effect of input scaling on sharpness and Jacobian norm for a ReLU network (5 trials). Line style denotes input scaling factor. Scaling data closer together does cause more severe increase in Jacobian norm, but corresponds to *less* severe increase in sharpness. This is due to the data scaling increasing the spectral norm of the feature maps.



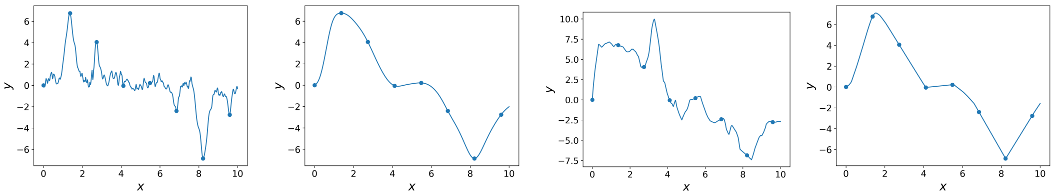
(a) Sharpness (b) Jacobian norm (c) Layer 1 feature norm (d) Layer 2 feature norm

Figure 31: Effect of input scaling on sharpness and Jacobian norm for a tanh network (5 trials). Line style denotes input scaling factor. Scaling data closer together does cause more severe increase in Jacobian norm, but corresponds to *less* severe increase in sharpness. This is due to the data scaling increasing the spectral norm of the feature maps.

C.3 THE ABSENCE OF THE FIRST LAYER JACOBIAN

We replicate the experiments of Section 8 in Ramasinghe et al. (2023), where it is demonstrated that flatness of minimum is not correlated with model smoothness in a simple regression task, and point to a possible explanation of this within our theory.

In detail, a four layer MLP with either Gaussian or ReLU activations is trained to regress 8 points in $\mathbb{R}^2$. Each network is trained from a *high frequency* initialisation (obtained from default PyTorch initialisation on the Gaussian network, and by pretraining on $\sin(6\pi x)$ for the ReLU network) to yield a non-smooth fit of the target data, and a *low frequency* initialisation (a wide Gaussian distribution for the Gaussian-activated network, and the default PyTorch initialisation for the ReLU network) to achieve a smooth fit of the data (see Figure 32). Refer to `coordinate_network.py` in the supplementary material for the code we used to run this experiment.



(a) Gaussian, high freq. (b) Gaussian, low freq. (c) ReLU, high freq. (d) ReLU, low freq.

Figure 32: Interpolations of 8 points by networks started from high frequency and low frequency initialisations. The smoother interpolations have lower Jacobian norm over the training data.

The models were trained with gradient descent using a learning rate of $1e-4$ and momentum of 0.9, for 10000 epochs for the Gaussian networks and 100000 epochs for the ReLU networks. The pretraining for the high frequency ReLU initialisation was achieved using Adam with a learning rate of $1e-4$ and default PyTorch settings.

Table 1: Norm values for regression networks, replicating result of Ramasinghe et al. (2023)

	Gaussian activated network	
Norm	High freq.	Low freq.
Jacobian norm	326.80 ± 182.51	84.07 ± 49.30
sharpness	13517.61 ± 3871.97	18353.54 ± 5751.98
First layer weight norm	9.09 ± 0.34	0.56 ± 0.02

Tables 1 and 2 record the means and standard deviations of loss Hessian and Jacobian norms of the Gaussian and ReLU networks over 10 trials. As in Ramasinghe et al. (2023), we find that while

Table 2: Norm values for regression networks, replicating result of Ramasinghe et al. (2023)

	ReLU activated network	
Norm	High freq.	Low freq.
Jacobian norm	121.59 ± 80.79	42.96 ± 7.82
sharpness	14211.05 ± 4767.26	9922.45 ± 3330.37
First layer weight norm	9.61 ± 0.28	9.56 ± 0.16

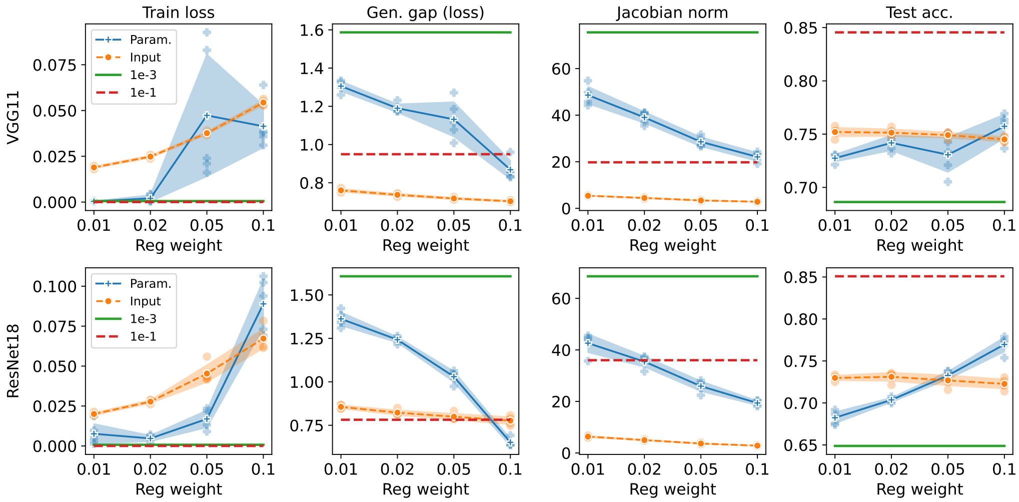
the smoothly interpolating ReLU models on average land in flatter minima than the non-smoothly interpolating models, the opposite is true for the Gaussian networks.

Keeping in mind that the high variances in these numbers make it difficult to come to a conclusion about the trend, we propose a possible explanation for why the Gaussian activated networks do not appear to behave according to Ansatz 3.1. From Equation (4), the Hessian cannot be expected to relate to the Jacobian of the first layer. Note now the discrepancy between the first layer weight norms in the low frequency versus high frequency fits with the Gaussian networks, which does not occur for the ReLU networks. We believe this to be the cause of the larger Hessian for the low-frequency Gaussian models: with such a small weight matrix in the first layer, all higher layers must be relied upon to fit the data, making their Jacobians higher than they would otherwise need to be. These larger Jacobians would then push the Hessian magnitude higher by Equation (4).

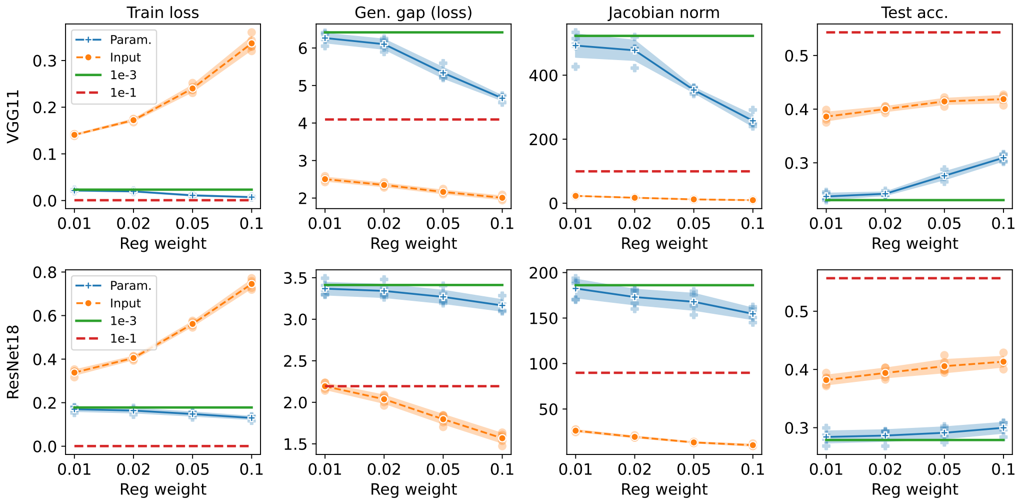
D RELATIONSHIP TO THE PARAMETER-OUTPUT JACOBIAN

In Lee et al. (2023), the *parameter-output* Jacobian is advocated as the mediating link between loss sharpness and generalisation. Extensive experiments are provided to support this claim. In particular, it is shown that regularising by the Frobenius norm of the parameter-output Jacobian during training is sufficient to alleviate the poor generalisation of (toy) networks trained using a small learning rate. However, no theoretical reasons are given for why the parameter-output Jacobian should be related to generalisation.

We expect that our Theorem 6.1 could provide the theoretical link missing in Lee et al. (2023). Indeed, the term in brackets in our Equation (4) is precisely the Gram matrix of the parameter-output Jacobian. Thus, our Ansatz 3.1 states that regularisation of the parameter-output Jacobian will also regularise the input-output Jacobian. If this is true, then we should see similar improvements in network performance when training with a small learning rate using an *input-output* Jacobian regulariser to those seen in Lee et al. (2023) when training with the *parameter-output* Jacobian regulariser. Moreover, one should see that these improvements are correlated to input-output Jacobian norm in all cases. Figure 33 demonstrates that this is indeed the case, using the same settings on data and models as were used for the regulariser experiments in Section 6.



(a) CIFAR10



(b) CIFAR100

Figure 33: All models trained for 90 epochs. x-axis denotes regularisation coefficient, and "Jacobian norm" refers to spectral norm. "ref 1e-1" and "ref 1e-3" are means over 5 trials of unregularised networks (regularisation coefficient set to zero) trained at learning rates of 1e-1 and 1e-3 respectively, shown for reference. The unregularised models trained with learning rate 1e-1 had their learning rate decayed by a factor of 10 every 30 epochs, so that their final learning rates were 1e-3. Learning rate scheduling was not used for any other trials."Param." is trained with the parameter-output regulariser of Lee et al, while "Input" is trained by regularising the minibatch Jacobian Frobenius norm using a similar approximation as Lee et al, namely $\|J\|_F \approx \|u^T J\|_2$ for u sampled uniformly from the $NC - 1$-sphere, where N is the batch size and C is the number of channels. 5 trials shown for the regularised trials, as well as means and 1 standard deviation shaded. Note the consistent downwards trend of both Jacobian norm and generalisation gap in all cases, as is consistent with our theory. Small Jacobian norm with large generalisation gap suggest, according to our theory, that the Jacobian of the model has ceased to be a good approximation of the Lipschitz constant of the model.