

Projet Deep Learning – Uncertainty Exploration

Rapport

Antoine Guidon
Fabio Tocco
Yelman Yahi
Anis Ouedghiri
Ram Nader

Table des matières

Introduction du projet	2
1 État de l’art et revue de l’article : Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles	3
2 Définition du protocole expérimental	5
2.1 Normalisation :	5
2.2 Périmètre d’étude et principe de comparabilité	6
2.3 Fonction de coût et optimiseur	6
2.4 Batch size, epochs, weight decay	6
2.5 Seeds, <code>shuffle</code> et initialisation — pourquoi les traiter séparément	6
2.6 Suivi des entraînements et prévention de l’overfitting	6
2.7 Tableau récapitulatif des hyperparamètres	7
3 Expérience 1 : Impact du pré-entraînement sur l’apprentissage de ResNet18	7
4 Expérience 2 : Robustesse au bruit gaussien	8
5 Expérience 3 : Effet d’un ensemble (7 modèles) sous bruit gaussien	9
6 Expérience 4 : Comportement hors distribution (MNIST \rightarrow KMNIST)	10
7 Expérience 5 : Robustesse à la rotation des chiffres MNIST	11
Annexes visuelles – Figures des expériences	13

Introduction du projet

Dans le cadre du module de Deep Learning, nous avons mené le projet intitulé **Uncertainty Exploration**, dont l'objectif est d'analyser la **fiabilité des prédictions de réseaux de neurones** à travers la notion d'incertitude. L'idée est de comprendre comment la confiance d'un modèle varie lorsqu'on le confronte à des données perturbées ou hors distribution, et d'évaluer dans quelle mesure une approche par **ensembles de modèles** (*Deep Ensembles*) permet d'obtenir des prédictions plus stables et mieux calibrées.

Pour cela, nous avons entraîné et comparé plusieurs modèles **ResNet18** sur le jeu de données **MNIST**, en suivant un protocole rigoureux permettant de :

- contrôler l'influence de l'initialisation des poids (modèle entraîné depuis zéro, pré-entraîné sur ImageNet) ;
- étudier l'impact du paramètre **shuffle** lors de l'entraînement ;
- analyser la sensibilité du modèle à différents types de perturbations : bruit blanc, rotation et données hors distribution (KMnist) ;
- et enfin, observer l'évolution de la confiance moyenne et de la variance des prédictions lorsque plusieurs modèles sont combinés en ensemble.

Avant d'obtenir et d'interpréter les résultats, il a été nécessaire de définir avec précision les paramètres de nos expériences. Les sections suivantes présentent d'abord un résumé de l'article scientifique de référence (*Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles*), puis détaillent les **choix méthodologiques** que nous avons effectués — notamment la stratégie de normalisation, les hyperparamètres d'entraînement et le périmètre des facteurs étudiés. Cette étape a permis d'établir un protocole commun, garantissant que les comparaisons entre modèles soient **fiables, reproductibles et scientifiquement interprétables**.

1 État de l’art et revue de l’article : Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles

Contexte et problématique

Les réseaux de neurones profonds (DNN) atteignent des performances remarquables dans de nombreuses applications, mais ils demeurent **mal calibrés** : ils produisent souvent des prédictions très confiantes même lorsqu’elles sont incorrectes. Cette absence de mesure fiable de l’incertitude constitue une limite majeure pour les usages sensibles (médical, automobile, finance), où il est crucial de savoir *à quel point le modèle est sûr de lui*.

Les approches bayésiennes classiques cherchent à modéliser l’incertitude en estimant une distribution de probabilité sur les poids du réseau. Cependant, ces méthodes sont souvent **lourdes à implémenter, coûteuses en calcul et difficiles à mettre à l’échelle**. L’article *Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles* propose une alternative non bayésienne, simple à appliquer et adaptée aux architectures modernes.

Approche proposée : les *Deep Ensembles*

L’idée centrale est d’obtenir des estimations d’incertitude fiables en combinant plusieurs réseaux de neurones entraînés indépendamment. Chaque réseau apprend une distribution de sortie, et leurs prédictions sont ensuite agrégées. L’approche repose sur trois composantes principales :

(a) Utilisation de règles de scoring appropriées. Chaque réseau prédit non seulement une valeur, mais une distribution de probabilité $p_{\theta}(y|x)$. En classification, on maximise la log-vraisemblance via la fonction *softmax*. En régression, le réseau prédit une moyenne $\mu(x)$ et une variance $\sigma^2(x)$, et l’on minimise la log-vraisemblance négative (NLL). Cette formulation permet au modèle d’apprendre explicitement la variance des prédictions et donc l’incertitude dite *aléatoire*.

(b) Entraînement adversarial. Pour lisser la surface de décision et renforcer la robustesse, les auteurs ajoutent des perturbations contrôlées aux entrées pendant l’apprentissage :

$$x' = x + \varepsilon \text{sign}(\nabla_x \ell(\theta, x, y))$$

où ε est une petite constante. Cette stratégie améliore la stabilité et la calibration des probabilités prédites en rendant le modèle moins sensible aux variations locales.

(c) Ensembles de réseaux indépendants. Les auteurs entraînent plusieurs réseaux ($M = 5$ à 10) avec des initialisations et des ordres de données différents. Chaque modèle est appris sur l’ensemble complet des données, puis les prédictions sont combinées :

$$p(y|x) = \frac{1}{M} \sum_{m=1}^M p_{\theta_m}(y|x)$$

Ce mélange de distributions capture l’*incertitude épistémique* (liée au modèle). L’entraînement indépendant rend la méthode naturellement **parallélisable**.

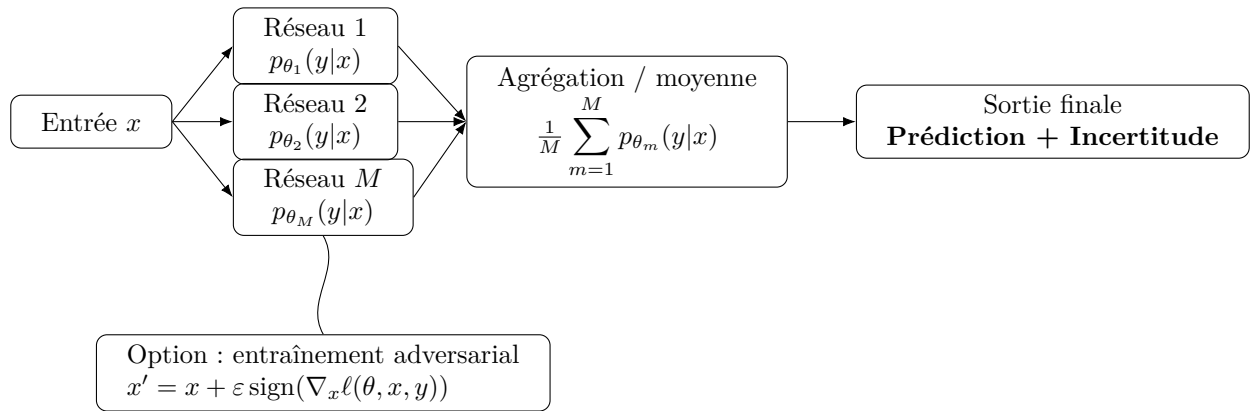


Figure 1 – Principe des *Deep Ensembles* : plusieurs réseaux indépendants produisent chacun une distribution prédictive $p_{\theta_m}(y|x)$; leurs sorties sont moyennées pour obtenir une prédiction globalement mieux **calibrée** et une estimation d'**incertitude**.

Résultats expérimentaux

Les auteurs valident leur méthode sur des tâches de **régression**, de **classification** et sur des cas de **données hors distribution**.

Régression. Sur les jeux de données standards (Boston Housing, Energy, Protein, etc.), les Deep Ensembles **égalent ou surpassent** les méthodes bayésiennes approximatives comme le *PBP* et le *MC-Dropout*. Ils obtiennent un **NLL plus faible**, gage d’une meilleure calibration, tandis que la précision (RMSE) reste similaire.

Classification et robustesse. Sur MNIST, SVHN et ImageNet, les ensembles améliorent la précision, la calibration (Brier score) et la robustesse face au bruit ou aux perturbations adversariales. Lorsqu’ils sont testés sur des données **hors distribution** (par ex. NotMNIST), ils attribuent une entropie plus élevée aux exemples inconnus, contrairement au MC-Dropout, souvent trop confiant. Appliquée à *ImageNet* avec le réseau *Inception*, la méthode conserve ces qualités à grande échelle, démontrant sa **scalabilité** sans modification architecturale.

Analyse et discussion

Cette approche se distingue par sa **simplicité**. Contrairement aux modèles bayésiens, elle ne requiert ni inférence postérieure ni ajustement de priors, tout en produisant des incertitudes fiables. Le coût d’entraînement croît avec le nombre de modèles, mais reste maîtrisé grâce au parallélisme.

Les Deep Ensembles capturent :

- l’incertitude **aléatoire**, liée au bruit des données ;
- l’incertitude **épistémique**, issue du désaccord entre réseaux.

Conclusion

Cet article montre qu’une approche non bayésienne simple peut fournir des incertitudes aussi fiables que les méthodes bayésiennes complexes. Les **Deep Ensembles** offrent un compromis idéal entre **précision**, **robustesse** et **simplicité** :

- prédictions bien calibrées ;
- robustesse au bruit et aux données hors distribution ;
- intégration directe dans les architectures modernes.

2 Définition du protocole expérimental

Avant de présenter les résultats des différentes expériences, il a été nécessaire de définir un cadre expérimental clair et reproductible. Nous avons donc :

- identifié les paramètres à **fixer** pour assurer la comparabilité entre tous les modèles (architecture, fonction de coût, optimiseur, taille de batch, etc.) ;
- sélectionné les paramètres à **explorer** dans nos ablations (initialisation des poids, **shuffle**, pré-entraînement, nombre de modèles dans un ensemble, etc.) ;
- choisi une méthode de normalisation cohérente avec notre domaine (MNIST) ;
- mis en place un suivi systématique des entraînements pour prévenir et détecter tout risque d’overfitting.

Ces choix structurent l’ensemble des expériences décrites dans la suite du rapport et garantissent que les écarts de performances observés proviennent bien des *facteurs étudiés* et non d’un biais de configuration.

2.1 Normalisation :

Soit une image $x \in [0, 1]^{H \times W}$ (MNIST, 1 canal), répliquée sur 3 canaux pour ResNet18. La normalisation affine appliquée avant le réseau s’écrit, canal par canal :

$$\tilde{x} = \frac{x - \mu}{\sigma}.$$

Nous avons envisagé trois stratégies :

1. **Norme ImageNet** : (μ, σ) fixés à ceux d’ImageNet.
2. **Norme MNIST** : (μ, σ) estimés sur le *train* MNIST (puis répliqués sur 3 canaux).
3. **Aucune normalisation** (ou simple mise à l’échelle $[0, 1]$).

Décision. Nous retenons la **normalisation MNIST** pour **tous** les entraînements (scratch et pré-entraîné), pour trois raisons principales :

- **Cohérence distributionnelle.** Les statistiques d’ImageNet sont inadaptées au domaine MNIST (grayscale, fond sombre/clair très spécifiques). Utiliser μ, σ ImageNet induit un *décalage de distribution artificiel* (covariate shift) *avant même* la première convolution, ce qui biaise l’interprétation des courbes de convergence et la comparabilité des variantes.
- **Comparaison équitable scratch vs pré-entraîné.** Notre objectif est d’isoler l’effet *du pré-entraînement* et *des hyperparamètres*, pas l’effet d’un choix de normalisation qui favoriserait mécaniquement l’une des variantes. En fixant la même normalisation **propre à MNIST** des deux côtés, on rend l’attribut « normalisation » neutre et on attribue les écarts de performance aux *vrais facteurs* étudiés (initialisation, shuffle, etc.).
- **Stabilité optimisation & interprétabilité.** La normalisation aux statistiques du dataset cible centre les activations d’entrée dans des plages adaptées à l’optimiseur et aux non-linéarités. Cela limite les effets de saturation précoces et réduit la variance d’entraînement d’un run à l’autre, ce qui rend nos *analyses d’incertitude* (et d’ensembles) plus lisibles.

Objection classique (pré-entraîné). On pourrait penser que le *pré-entraîné* “attend” l’échelle ImageNet. En pratique :

- la tête de classification est remplacée
- la réplique grayscale \rightarrow RGB modifie déjà les statistiques d’entrée
- ResNet18 reste robuste à une *affine rescaling* raisonnable des entrées.

Empiriquement, on observe une convergence tout à fait stable avec une normalisation MNIST, et la comparaison scratch/pré-entraîné devient **plus honnête**.

2.2 Périmètre d'étude et principe de comparabilité

Pour **toutes** les expériences, nous fixons des **hyperparamètres communs** (fonction de coût, optimiseur, lr, weight decay, batch size, nombre d'épochs, split train/val, etc.) afin de garantir que les différences observées proviennent *des facteurs étudiés* (initialisation, **shuffle**, ensembles, corruptions...). Nous verrouillons aussi la **graine aléatoire** pour la phase d'ingénierie/validation, et nous la **déverrouillerons** lorsque des runs indépendants sont nécessaires (p.ex. ensembles de 7 modèles).

2.3 Fonction de coût et optimiseur

- **Loss : CrossEntropyLoss**. Cible unique (10 classes), probabilités calibrées via softmax, signal de gradient stable et largement éprouvé.
- **Optimiseur : Adam**. Accélère le warm-up sur un domaine simple (MNIST) tout en restant robuste aux variations d'échelle des gradients. C'est un bon *par défaut* pour comparer des variantes d'initialisation/seed sans introduire un bruit d'optimisation excessif.

2.4 Batch size, epochs, weight decay

- **Batch size** : choisi pour saturer raisonnablement le CPU/GPU et lisser le bruit de gradient (*ici* 512).
- **Epochs** : budget court (p.ex. 3) pour les premiers diagnostics de *tendance* ; augmenté ensuite si nécessaire pour atteindre la borne qualité $\geq 90\%$ demandée.
- **Weight decay** : régularisation ℓ_2 modérée pour limiter l'overfitting dans les runs rapides et rendre les comparaisons plus stables.

2.5 Seeds, shuffle et initialisation — pourquoi les traiter séparément

- **Seed globale** : garantit la reproductibilité exacte (tirages de batches, split train/val, initialisation pseudo-aléatoire, etc.). *On la fixe pour l'ingénierie*, puis **on la varie** quand on a besoin d'*indépendance* (ensembles).
- **shuffle** : affecte le *chemin* d'optimisation (ordre des gradients). Nous testons son impact séparément (ON vs OFF) car il peut interagir avec la seed et l'initialisation.
- **Initialisation des poids** : trois cas pertinents dans notre protocole :
 1. *default* (PyTorch) — baseline robuste,
 2. *pré-entraînement ImageNet* — transfert de représentation,
 3. *réinitialisations contrôlées* (*Kaiming, Xavier*) — pour isoler l'impact de l'init seule à capacité constante.

2.6 Suivi des entraînements et prévention de l'overfitting

Afin de nous prémunir contre l'overfitting et de garantir que les résultats présentés reflètent un apprentissage réellement généralisable, chaque modèle entraîné a fait l'objet d'un **suivi automatisé** des performances :

- À chaque epoch, les valeurs de **train loss**, **train accuracy**, **validation loss** et **validation accuracy** sont enregistrées dans un fichier `metrics.csv` à l'aide de la classe `MetricsTracker` (voir `tools.py`).
- Les courbes correspondantes sont automatiquement tracées et sauvegardées (`loss_curves.png` et `accuracy_curves.png`).
- Le modèle sauvegardé correspond toujours à **l'état de validation le plus performant** (loss minimale), afin d'éviter de retenir une itération surapprise.

Ces courbes permettent de vérifier visuellement la bonne convergence des modèles : une divergence progressive entre la loss d'entraînement et celle de validation signalerait un début d'overfitting. Aucun modèle n'a été conservé sans que cette vérification soit effectuée.

2.7 Tableau récapitulatif des hyperparamètres

Table 1 – Hyperparamètres utilisés : **fixés** pour la comparabilité vs **à explorer** dans nos ablations.

Catégorie	Nom	Statut	Justification / Plan d'analyse
Données	Normalisation	Fixé = MNIST	Voir §2.1. Évite un biais en faveur de la variante pré-entraînée et stabilise l'optimisation.
Données	Taille / Canaux	Fixé = 32×32, RGB	Compatibilité ResNet18; réplication grayscale→RGB.
Split	Train/Val/Test	Fixé = 80/20 / officiel	Val contrôlée pour suivi; test intact pour reporting.
Perfs	Seuil qualité	Fixé = $\geq 90\%$	Exigence minimale du sujet (modèle pertinent).
Modèle	Archi	Fixé = ResNet18	Compromis capacité/temps d'entraînement.
Modèle	Tête	Fixé = Linear(512,10)	Adaptation 10 classes.
Optim	Loss	Fixé = CrossEntropy	Standard multi-classes, stable.
Optim	Optimiseur	Fixé = Adam	Warm-up rapide, variance modérée entre runs.
Optim	LR	Fixé = 1×10^{-4}	Suffisant pour convergence stable dans nos budgets courts.
Optim	Weight decay	Fixé = 1×10^{-4}	Régularisation modérée, limite l'overfit.
Optim	Batch size	Fixé = 512	Lissage du bruit de gradient / débit calcul.
Budget	Epochs	Fixé (diag) = 20	Diagnostics rapides; on étendra si besoin.
Réplica	Seed globale	Fixé (ingénierie)	Reproductibilité stricte; sera variée pour les ensembles.

3 Expérience 1 : Impact du pré-entraînement sur l'apprentissage de ResNet18

Objectif et protocole

Cette première expérience vise à comparer deux stratégies d'initialisation pour un même modèle **ResNet18** appliqué à MNIST :

- une version **entraînée depuis zéro** (poids aléatoires – “from scratch”);
- une version **pré-entraînée sur ImageNet**, dont seule la couche finale (**fc**) est réinitialisée.

L'objectif est d'observer l'influence du pré-entraînement sur la vitesse de convergence, la stabilité de l'apprentissage et la performance finale. Les deux entraînements ont été effectués avec la même configuration expérimentale afin de garantir une comparaison équitable :

- normalisation : **MNIST** (moyenne et écart-type estimés sur le train uniquement);
- nombre d'epochs : 20;
- taille de batch : 512;
- optimiseur : Adam ($lr = 10^{-4}$, $wd = 10^{-4}$);
- fonction de perte : CrossEntropyLoss;
- seed fixe = 0, **shuffle** = False.

Résultats

Les courbes d'évolution de la perte et de l'accuracy (voir Figures 2a–2d) montrent des différences nettes entre les deux approches :

- Le modèle **pré-entraîné** démarre avec une perte beaucoup plus faible et atteint rapidement une accuracy supérieure à 95 % dès les premières epochs. La convergence est stable et la différence entre train et validation reste faible, signe d'un bon compromis biais-variance.
- Le modèle **from scratch** affiche une perte initiale plus élevée et une progression plus lente. L'accuracy augmente progressivement jusqu'à environ 92–93 %, avec une courbe de validation

plus bruitée.

Ces écarts traduisent l'effet positif du pré-entraînement : les couches convolutives, déjà optimisées sur ImageNet, capturent rapidement des motifs génériques (bords, contrastes, formes simples) utiles pour MNIST, malgré la différence de domaine. Ainsi, la phase d'optimisation se concentre essentiellement sur l'adaptation des représentations haut niveau plutôt que sur l'apprentissage des filtres de bas niveau.

Analyse comparative

Sur les 20 epochs d'entraînement, la **vitesse de convergence** du modèle pré-entraîné est environ **3 à 4 fois plus rapide** : dès l'epoch 5, sa perte de validation est déjà inférieure à celle du modèle scratch en fin de training. De plus, la stabilité des courbes d'accuracy montre que le pré-entraînement agit comme une **forme implicite de régularisation**, en réduisant la variance d'initiation et les oscillations d'apprentissage.

Ces observations confirment les attentes théoriques :

- le pré-entraînement apporte un *gain de généralisation* sur des tâches simples, même avec un dataset de petite taille ;
- il permet de réduire le coût de calcul en diminuant le nombre d'epochs nécessaires pour atteindre le seuil de $> 90\%$ de l'accuracy.

Synthèse de l'expérience 1

- Le pré-entraînement sur ImageNet améliore nettement la vitesse de convergence et la performance initiale.
- Le modèle from scratch reste capable d'atteindre une bonne précision, mais nécessite davantage d'epochs pour stabiliser son apprentissage.

4 Expérience 2 : Robustesse au bruit gaussien

Objectif et protocole

Cette expérience évalue la **robustesse du modèle face à un bruit gaussien additif** appliqué directement aux pixels des images MNIST. L'objectif est d'observer comment la dégradation progressive du signal d'entrée affecte les prédictions du réseau, tant en termes de justesse que de confiance.

Le bruit est simulé sous la forme d'un **bruit blanc gaussien additif**, caractérisé par un paramètre d'intensité noté `level`. Ce paramètre contrôle l'amplitude du bruit injecté : un `level` de 0.1 correspond à une perturbation légère à peine visible, tandis qu'un `level` de 0.5 produit une forte altération des pixels, rendant certaines images difficilement lisibles même pour un humain.

Les tests ont été effectués à cinq niveaux de bruit : 0.1, 0.2, 0.3, 0.4 et 0.5. Le modèle utilisé est le même **ResNet18 entraîné from scratch sur MNIST** que dans l'expérience précédente, avec les hyperparamètres suivants :

- `pretrained = False`, `epochs = 20`, `batch_size = 512`;
- `learning_rate = 1e-4`, `weight_decay = 1e-4`, `optimizer = Adam`;
- fonction de coût : `CrossEntropyLoss` ; normalisation : MNIST.

Analyse qualitative

Les Figures 3a–3f présentent un sous-ensemble fixe de chiffres du jeu de test soumis à différents niveaux de bruit.

Images propres (Figure 3a). Sans bruit, le modèle identifie correctement la totalité des chiffres avec une confiance supérieure à 99 %, confirmant une excellente performance de base.

Bruit faible (niveau = 0.1, Figure 3b). La classification reste correcte pour la quasi-totalité des exemples. Seules quelques confusions ponctuelles apparaissent (par exemple 9→8), sans impact significatif sur la confiance globale.

Bruit modéré (niveaux 0.2–0.3, Figures 3c–3d). La qualité des prédictions commence à se dégrader : la probabilité associée à la classe correcte diminue, et des inversions de classes apparaissent, notamment entre chiffres aux formes similaires (58, 14). Le modèle conserve une apparente confiance, ce qui traduit une **mauvaise calibration** face à la dégradation du signal.

Bruit fort (niveaux 0.4–0.5, Figures 3e–3f). Les erreurs deviennent fréquentes et systématiques. Le réseau prédit souvent la même classe (« 1 ») pour des entrées fortement bruitées, signe que la distribution des activations internes est saturée par le bruit. Cette perte de diversité dans les sorties s’accompagne d’une chute nette de la fiabilité du modèle.

Interprétation

Cette expérience met en évidence une **sensible vulnérabilité du modèle aux perturbations additives**. Bien que performant sur des images propres, le réseau échoue à maintenir des représentations stables dès que le bruit dépasse un certain seuil. Le phénomène de surconfiance persiste même lorsque les images deviennent méconnaissables, illustrant une calibration défaillante du softmax.

D’un point de vue applicatif, cette expérience montre la nécessité d’introduire soit une **data augmentation bruitée** lors de l’apprentissage, soit une approche plus robuste telle que les *ensembles* qu’on utilisera juste après (*cf. expérience 3*).

Synthèse de l’expérience 2

- Le modèle conserve de bonnes performances jusqu’à un niveau de bruit < 0.2 , mais se dégrade fortement au-delà.
- Les scores de confiance demeurent élevés même en cas d’erreurs, révélant une surconfiance structurelle.
- Cette expérience confirme le besoin d’évaluer la robustesse et la calibration des modèles à travers des perturbations contrôlées.

5 Expérience 3 : Effet d’un ensemble (7 modèles) sous bruit gaussien

Objectif et protocole

On étudie si un **ensemble de 7 modèles** (mêmes hyperparamètres, seeds différentes) améliore la **robustesse au bruit** par rapport à un *single* (meilleur modèle). Le bruit est additif gaussien (zéro-moyenne) d’écart-type $\sigma \in \{0.0, 0.2, 0.3, 0.4, 0.5\}$ injecté sur les pixels *après* normalisation MNIST. On évalue visuellement 20 images fixes (grilles des Figures 4–8) et quantitativement l’**accuracy** sur ces 20 images (Figure 9).

Résultats

Qualitatif (grilles). Jusqu’à $\sigma = 0.2$, les deux approches restent très fiables : les chiffres restent lisibles et les prédictions sont correctes (*cf.* Figure 5). Dès $\sigma = 0.3$ –0.4, des confusions apparaissent (formes « cassées », traits rompus) ; l’ensemble conserve des scores plus **cohérents** (moins d’extrêmes) que le *single* (*cf.* Figures 6–7). À $\sigma = 0.5$, les erreurs deviennent fréquentes et la confiance du *single*

est souvent trop élevée sur des prédictions fausses, alors que l'ensemble **abaisse** davantage ses probabilités (meilleure calibration visuelle), Figure 8.

Quantitatif (courbe). La Figure 9 montre que l'ensemble domine systématiquement le *single* quand le bruit augmente : à $\sigma = 0.4$, l'ensemble atteint $\sim 85\%$ d'accuracy (vs $\sim 75\%$ pour le *single*) ; à $\sigma = 0.5$, l'écart reste net ($\sim 65\%$ vs $\sim 60\%$). L'avantage est faible à bas bruit (effet *ceiling*), puis **croît avec la sévérité** des corruptions.

Interprétation

L'**agrégation** (moyenne des softmax) agit comme une *régularisation* : elle réduit la variance inter-modèles, limite les sur-confiances idiosyncratiques et fournit des scores plus stables quand le signal se dégrade. Cependant, l'ensemble ne crée pas d'invariance structurelle au bruit : au-delà d'un certain seuil, les formes sont trop altérées et la performance chute pour toutes les méthodes.

Synthèse

- À bruit faible, *single* et *ensemble* sont presque équivalents (effet plafond).
- À bruit modéré/fort, l'ensemble garde **plus d'accuracy** et une **meilleure calibration apparente**.
- Les ensembles atténuent la variance mais ne résolvent pas le problème quand l'information visuelle disparaît.

6 Expérience 4 : Comportement hors distribution (MNIST \rightarrow KMNIST)

Objectif et protocole

Cette expérience vise à étudier la capacité du modèle à généraliser **hors de la distribution d'entraînement (OOD)**. L'idée est d'évaluer dans quelle mesure un modèle entraîné sur MNIST conserve une cohérence prédictive lorsqu'il est confronté à un jeu de données très différent mais de même format, **KMNIST** — un ensemble d'images représentant des caractères japonais (*hiragana*).

Nous utilisons le même **ResNet18 entraîné from scratch sur MNIST** (20 époques, `batch_size` = 512, `lr` = `1e-4`, Adam, `CrossEntropyLoss`). Deux ensembles d'images sont présentés :

- **MNIST** : 20 images de chiffres utilisées en référence (Figure 10a) ;
- **KMNIST** : 20 images provenant du dataset hors distribution (Figure 10b).

Les annotations au-dessus de chaque image indiquent :

- y : la vraie étiquette ;
- S : prédiction du modèle seul (*Single model*) avec sa probabilité ;
- E : prédiction moyenne issue de l'*Ensemble* de 7 modèles indépendants.

Analyse des résultats

MNIST (in distribution). Sur MNIST (Figure 10a), le modèle reproduit les performances observées précédemment : la quasi-totalité des prédictions sont correctes avec une confiance supérieure à 99 %. L'ensemble de modèles (*ensemble averaging*) ne modifie que marginalement les sorties, confirmant la stabilité du réseau sur des données connues.

KMNIST (hors distribution). En revanche, sur KMNIST (Figure 10b), le comportement change radicalement :

- Les chiffres n'ayant aucune correspondance sémantique avec les caractères japonais, les prédictions sont erronées dans plus de 90 % des cas.

- Malgré cela, les probabilités associées demeurent souvent **supérieures à 70–90 %**, montrant une forte surconfiance du modèle sur des échantillons complètement étrangers.
- L’ensemble de modèles (E) produit des scores de confiance légèrement plus faibles et plus cohérents, mais il ne corrige pas fondamentalement les erreurs de classification.

Interprétation

Ces observations mettent en évidence une **mauvaise calibration hors distribution**. Le modèle interprète les motifs inconnus comme des chiffres qu’il connaît le mieux, sans mécanisme de doute explicite. L’ensemble (*ensemble averaging*) agit comme un *lissage probabiliste*, réduisant les extrêmes mais ne permettant pas une détection fiable des échantillons OOD.

Ce phénomène illustre la limite des réseaux entraînés de manière purement discriminative : la sortie softmax est contrainte de normaliser les probabilités sur les 10 classes apprises, même lorsque l’entrée n’appartient à aucune d’entre elles. Ainsi, le réseau doit « choisir » une classe malgré son ignorance, ce qui conduit à une **fausse confiance élevée sur des données inconnues**.

Synthèse de l’expérience 4

- Sur MNIST, le modèle et son ensemble restent parfaitement calibrés et cohérents.
- Sur KMNIST, les performances chutent drastiquement, mais les scores de confiance demeurent élevés : **forte surconfiance OOD**.
- L’ensemble de modèles réduit légèrement la variance des prédictions, mais ne permet pas une vraie détection des exemples hors distribution.
- Cette expérience souligne la nécessité de mesures d’incertitude plus fines (entropie, variance prédictive, ou score OOD explicite) pour fiabiliser les modèles déployés.

7 Expérience 5 : Robustesse à la rotation des chiffres MNIST

Objectif et protocole

Cette dernière expérience vise à évaluer la **sensibilité angulaire** des modèles entraînés sur MNIST. Nous cherchons à observer comment la probabilité prédite pour la vraie classe varie lorsque les chiffres sont progressivement tournés, de 0° à 180° , avec un pas de 10° .

Pour cela, nous avons réutilisé les **7 modèles** entraînés précédemment sur MNIST (mêmes hyperparamètres, seeds différentes). Chaque modèle a été appliqué sur un sous-ensemble aléatoire de **20 images de test** (voir Figure 11). Chaque image a ensuite été soumise à une rotation allant de 0° à 180° par pas de 10° .

L’analyse s’est déroulée en deux temps :

- observation des variations de probabilité sur un **seul modèle** pour évaluer la cohérence individuelle ;
- comparaison entre modèles (mêmes poids initiaux mais seeds différentes) afin de vérifier si l’**ensemble de 7 modèles** lisse ou non ces variations.

Analyse des résultats

Variation angulaire individuelle. La Figure 13a présente la probabilité associée à la **vraie classe** pour plusieurs exemples distincts (chiffres 1, 3, 8 et 9), prédite par un seul modèle (modèle 5). On observe que la probabilité chute brutalement dès que la rotation dépasse environ 30° , avant de remonter ponctuellement pour certaines orientations (autour de $120\text{--}150^\circ$). Ces fluctuations correspondent souvent à des symétries visuelles partielles (par exemple, un “3” renversé pouvant rappeler un “8”).

Ces oscillations traduisent le fait que le réseau n’a appris que des représentations très dépendantes de l’orientation. Il ne dispose donc pas d’une invariance géométrique réelle : le modèle reste sensible

à toute transformation affine absente du jeu d'entraînement.

Visualisation qualitative. La Figure 12 illustre le cas d'une image du chiffre "3" (idx 442). À 0° , le modèle est parfaitement confiant ($P = 1.00$), mais dès 45° il la classe comme un "7", puis comme un "0" ou un "8" à partir de 90° . Ce comportement montre que la décision du modèle repose essentiellement sur des motifs de contour locaux, sans compréhension globale invariante à la rotation.

Comparaison inter-modèles. Les courbes de la Figure 13a et 13b montrent les probabilités de la vraie classe pour plusieurs modèles indépendants (modèles 2 et 5). Les profils sont presque identiques : les chutes et remontées de confiance surviennent aux mêmes angles. Cela montre que l'aléa de l'initialisation n'a qu'un impact marginal : la sensibilité à la rotation est **structurelle** à l'architecture et non accidentelle.

Ensemble de modèles. La Figure 14 présente la moyenne des probabilités sur les sept modèles. L'effet d'ensemble agit comme une **opération de lissage** : les fluctuations locales sont atténuées, mais la tendance générale (perte de confiance rapide entre 30° et 90°) demeure. En d'autres termes, combiner plusieurs modèles améliore légèrement la stabilité numérique, mais ne confère pas d'invariance à la rotation.

Synthèse de l'expérience 5

- La rotation constitue une perturbation très sensible pour un réseau entraîné sur MNIST sans data augmentation.
- Tous les modèles (mêmes hyperparamètres, seeds différentes) réagissent de manière quasi identique : la sensibilité angulaire est donc **intrinsèque à l'architecture et aux données**, non à l'initialisation.
- L'ensemble de 7 modèles réduit légèrement la variance et rend les probabilités plus régulières, mais ne rétablit pas la justesse des prédictions au-delà de 40° – 60° .
- Ces résultats confirment que les ensembles de modèles **améliorent la stabilité et la calibration**, sans résoudre le manque d'invariance structurelle.

Conclusion générale

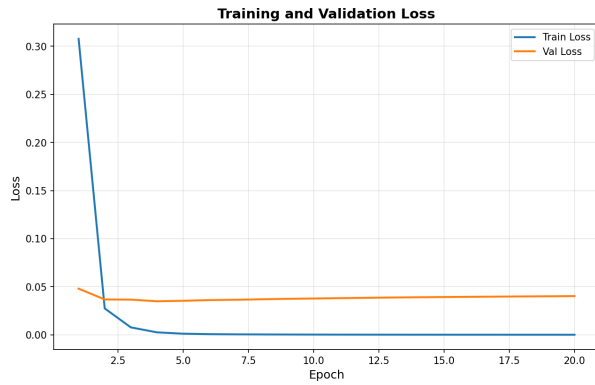
Ce projet nous a permis d'explorer concrètement la notion d'incertitude dans les réseaux de neurones profonds, en reproduisant et en adaptant les principes de l'article : "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles" sur les *Deep Ensembles*. À travers nos cinq expériences, nous avons mis en évidence les limites de modèles classiques comme ResNet18 face à des perturbations simples (bruit, rotation, données hors distribution) : bien que performants sur des données propres, ils demeurent fortement sur-confiants lorsque les entrées s'éloignent de la distribution d'entraînement.

L'utilisation d'ensembles de modèles indépendants a montré une amélioration notable de la stabilité et de la calibration des prédictions, confirmant les conclusions de l'article de référence. Les *Deep Ensembles* constituent ainsi une approche simple mais efficace pour capturer à la fois l'incertitude aléatoire (liée aux données) et l'incertitude épistémique (liée au modèle).

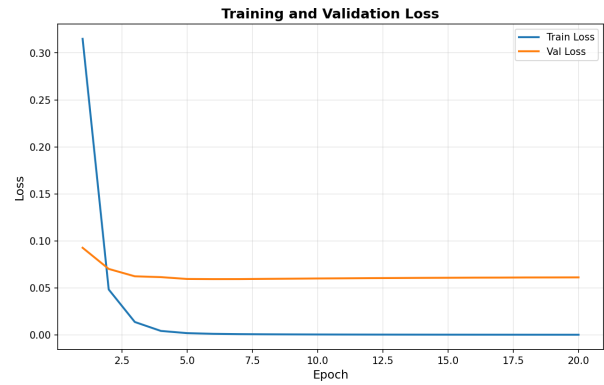
Néanmoins, nos résultats rappellent que cette méthode ne confère pas d'invariances structurelles au réseau : elle lisse les erreurs sans les éliminer. Une perspective naturelle serait d'associer les ensembles à des techniques d'augmentation de données ou à des architectures explicitement invariantes, afin d'obtenir des modèles à la fois robustes, calibrés et plus conscients de leur propre incertitude.

Annexes visuelles – Figures des expériences

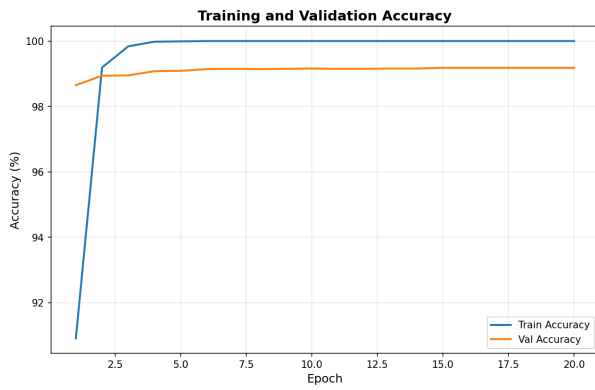
Expérience 1 – Courbes d'apprentissage (comparaison pré-entraîné / scratch)



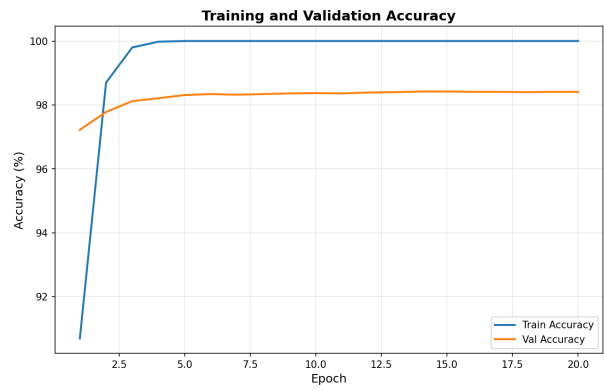
(a) Perte (train/val) – modèle pré-entraîné.



(b) Perte (train/val) – modèle scratch.



(c) Accuracy (train/val) – modèle pré-entraîné.



(d) Accuracy (train/val) – modèle scratch.

Figure 2 – Comparaison des courbes de perte et d'accuracy entre le modèle pré-entraîné et le modèle from scratch sur MNIST.

Expérience 2 – Robustesse au bruit gaussien (illustrations qualitatives)

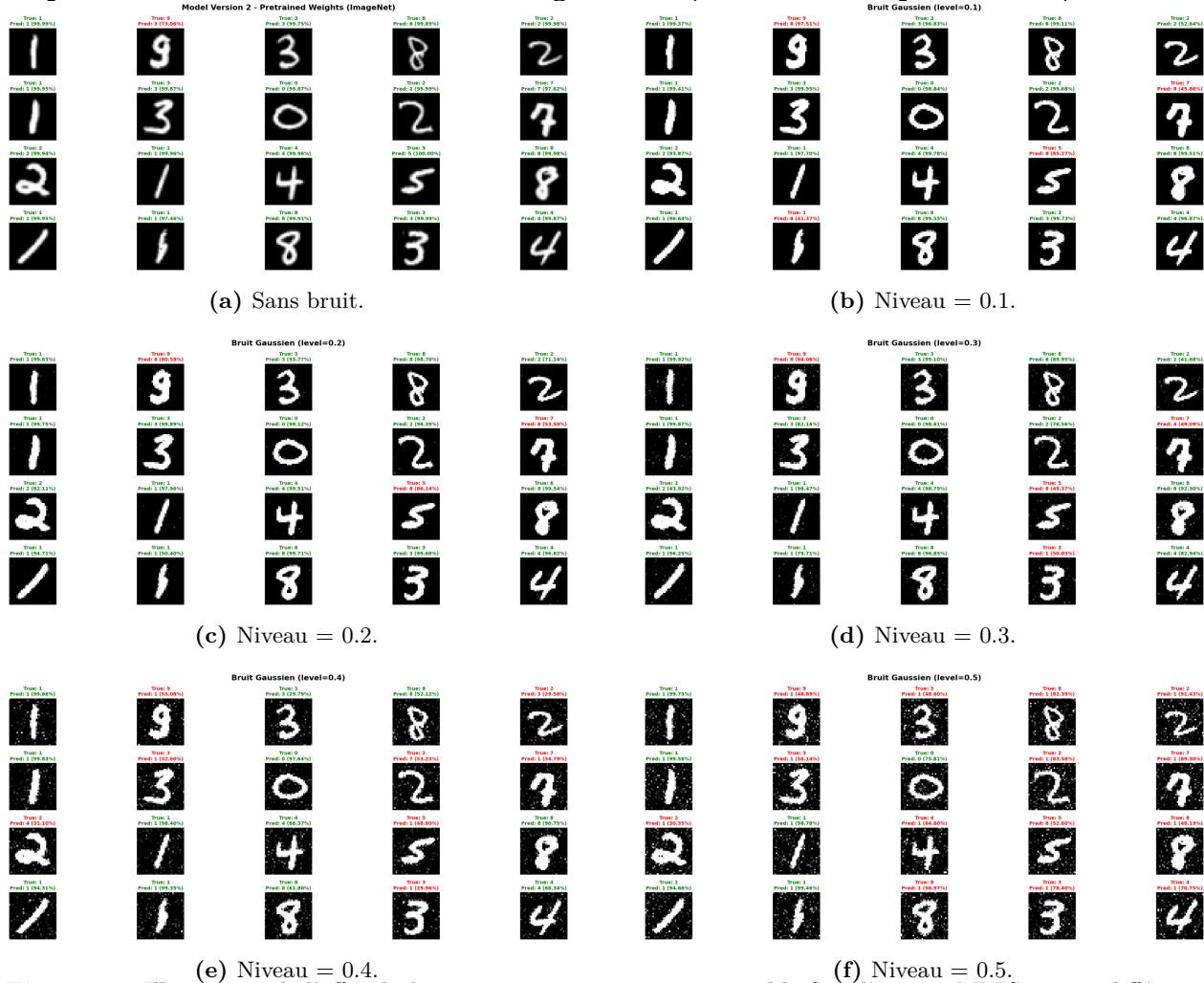


Figure 3 – Illustration de l’effet du bruit gaussien sur un sous-ensemble fixe d’images MNIST pour différents niveaux de perturbation.

Expérience 3 – Grilles qualitatives et courbe d'accuracy

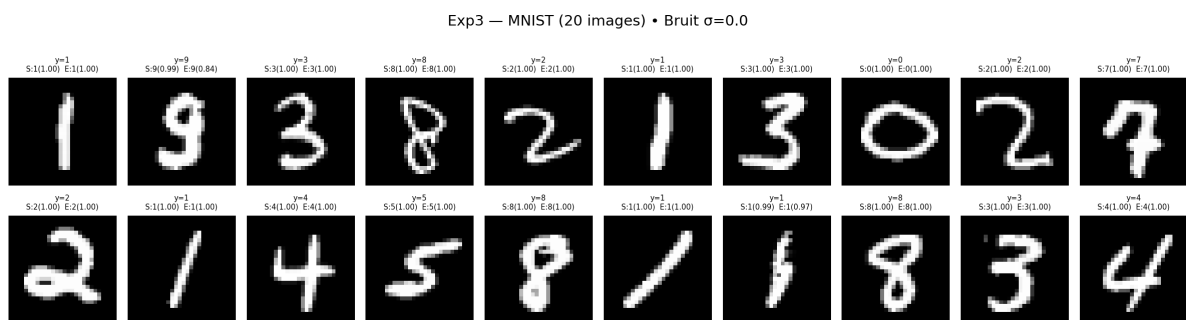


Figure 4 – MNIST sous $\sigma = 0.0$ (référence).

Exp3 — MNIST (20 images) • Bruit $\sigma=0.2$

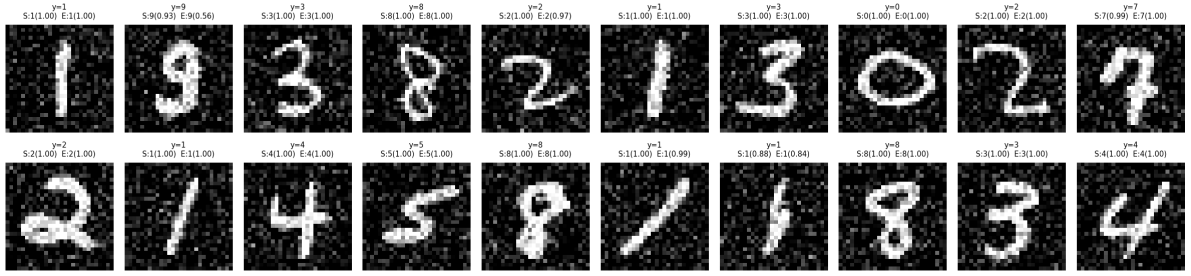


Figure 5 — $\sigma = 0.2$: prédictions encore stables.
Exp3 — MNIST (20 images) • Bruit $\sigma=0.3$

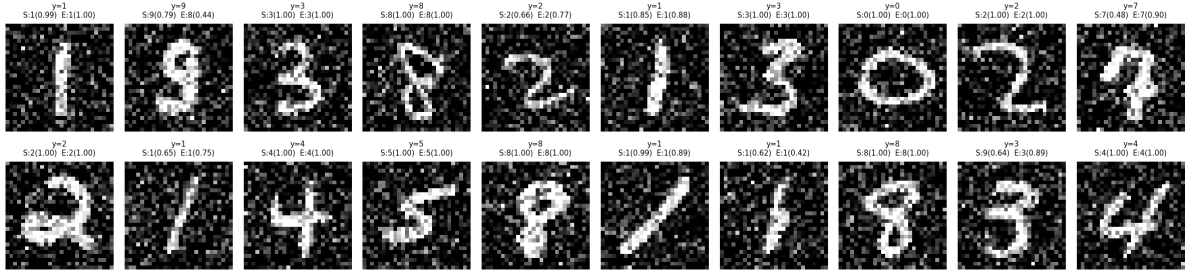


Figure 6 — $\sigma = 0.3$: premières confusions visibles.
Exp3 — MNIST (20 images) • Bruit $\sigma=0.4$

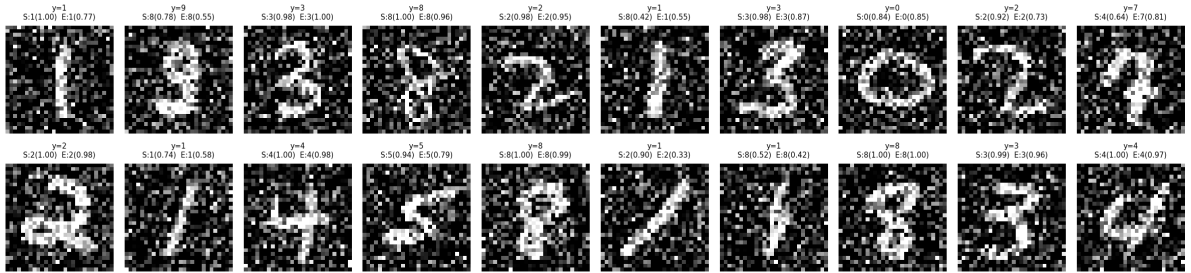


Figure 7 — $\sigma = 0.4$: l'ensemble reste mieux calibré.
Exp3 — MNIST (20 images) • Bruit $\sigma=0.5$

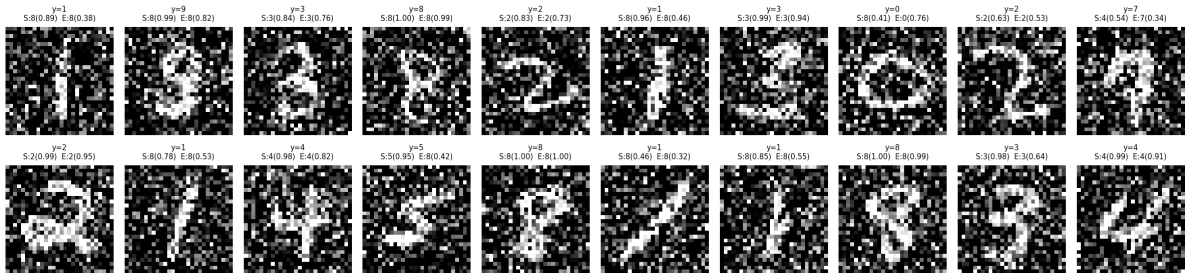


Figure 8 — $\sigma = 0.5$: forte dégradation, avantage résiduel de l'ensemble.

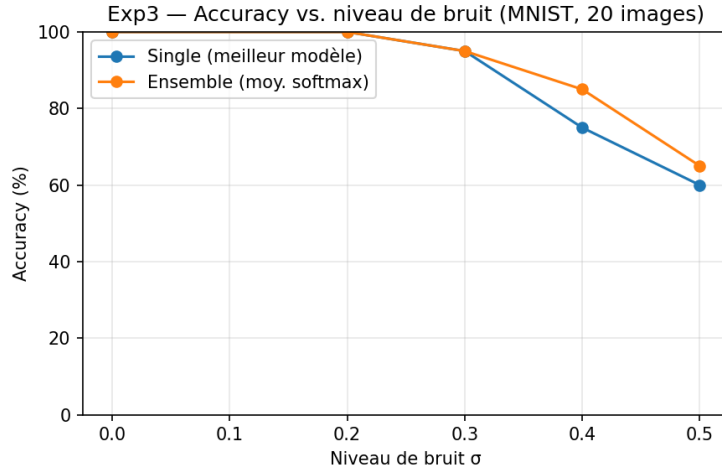
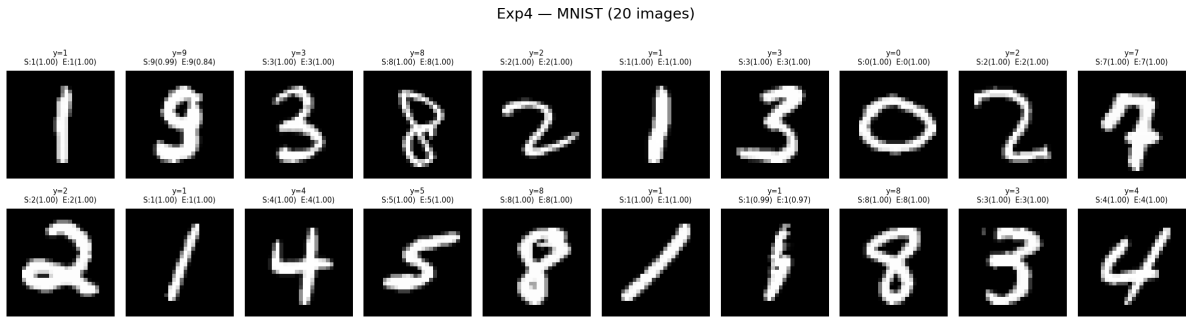
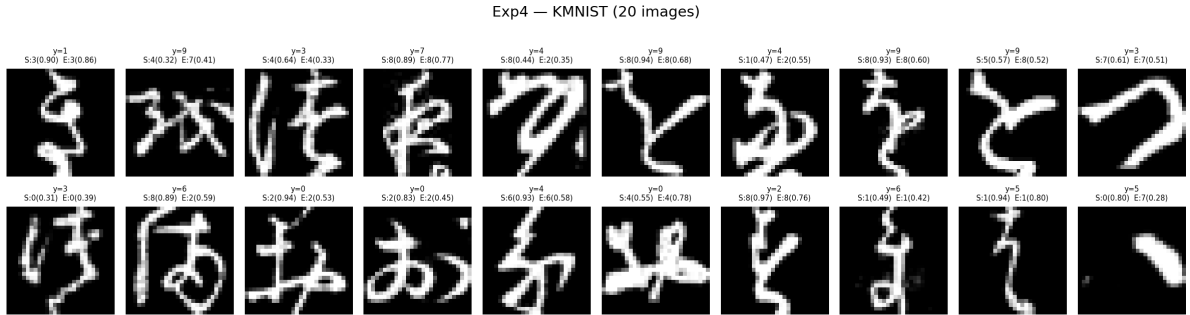


Figure 9 — Accuracy (%) selon σ — *Single* vs *Ensemble* (20 images fixes).

Expérience 4 — Illustrations qualitatives (MNIST vs KMNIST)



(a) Échantillons du dataset MNIST (20 images). Les prédictions du modèle seul (S) et de l'ensemble (E) sont identiques et correctes avec des confiances proches de 1.0.



(b) Échantillons du dataset KMNIST (20 images). Les prédictions sont aléatoires, souvent fausses, mais accompagnées de fortes confiances, illustrant la surconfiance hors distribution.

Figure 10 — Comparaison du comportement du modèle sur des images **in distribution** (MNIST) et **hors distribution** (KMNIST).

Expérience 5 – Robustesse à la rotation

MNIST — 20 images utilisées pour l'expérience

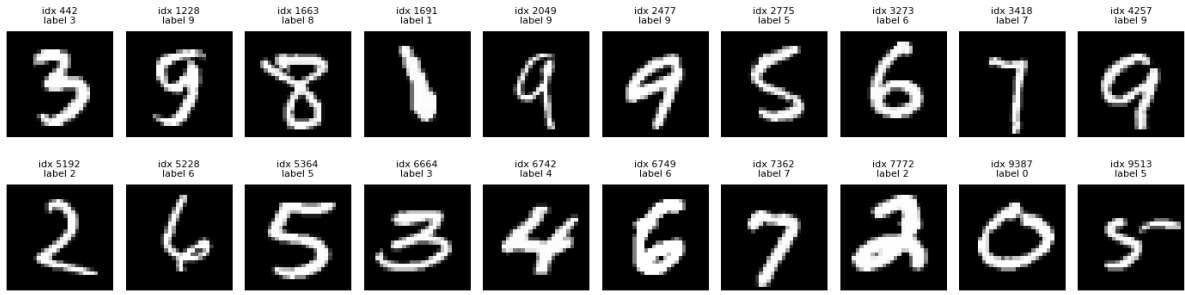


Figure 11 – Les 20 images de MNIST utilisées pour l'expérience 5 (tirage aléatoire du jeu de test).

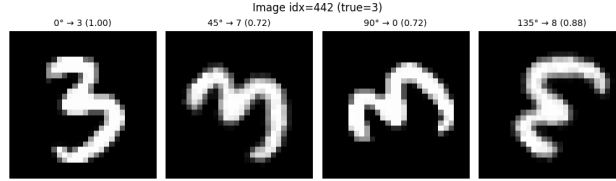
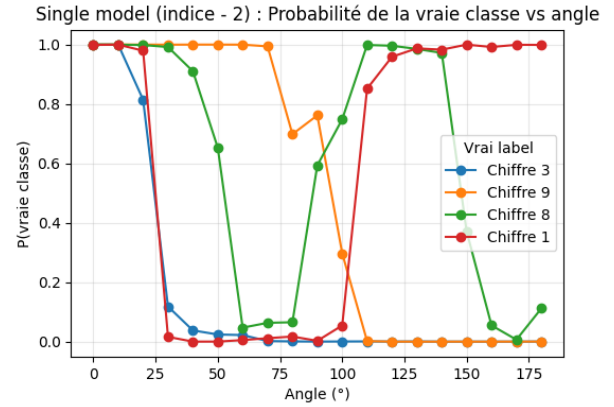
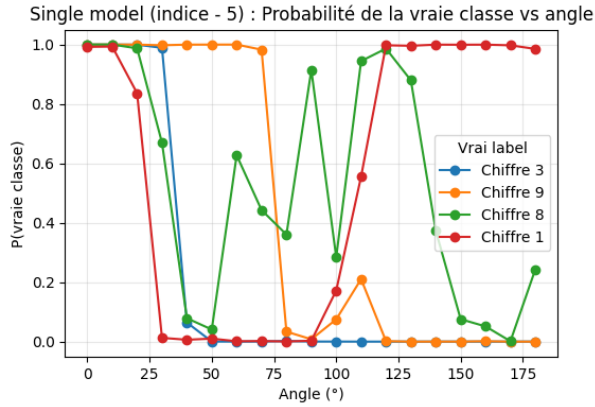


Figure 12 – Évolution des prédictions d'une même image ($true = 3$) selon l'angle de rotation.



(a) Modèle indice 5 : probabilité de la vraie classe vs angle.

(b) Modèles indice 2 : probabilité de la vraie classe vs angle.

Figure 13 – Comportement individuel des modèles face aux rotations : mêmes chutes de confiance entre 30° et 100°.

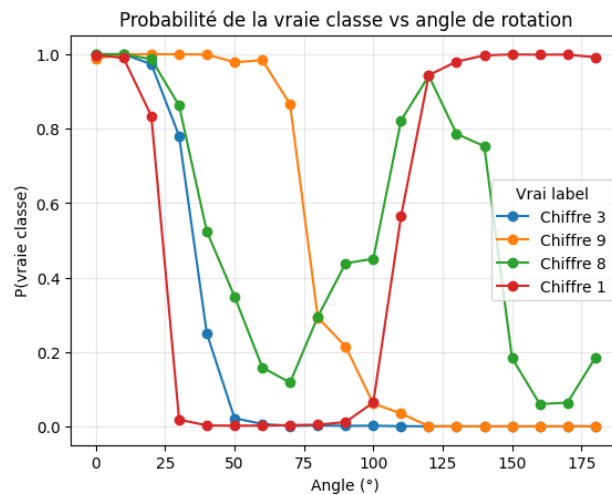


Figure 14 – Moyenne des probabilités de vraie classe sur les 7 modèles. L'ensemble lisse les variations mais ne crée pas d'invariance angulaire.