**Université Moulay Ismail**

# Reconnaissance de la Langue des Signes

---

**Filière :** SDIA M1
**Réalisé par :** DIABATE YOUSSOUF
**Encadrant :** Mr. Bekri & Ilham EL OUARICHIf

Année universitaire 2024–2025

# Table des matières

Première partie

# Partie I : Etude Contextuelle

## 0.1 Introduction :

Les langues des signes (LS) sont des langues naturelles pratiquées au sein des communautés de Sourds. Les LS sont des langues sans écriture, relevant, à ce titre, de la modalité orale, du face-àface, même transmis en différé. Elles sont visuo-gestuelles, produites par de nombreux articulateurs corporels, les mains et les bras bien sûr, mais aussi le visage, les épaules, la tête, le regard et plusieurs composantes faciales, qui peuvent être activés plus ou moins simultanément et perçues par les yeux. Cette modalité de perception est bien adaptée à l'interprétation de mouvements organisés dans l'espace. De fait, cet espace joue un rôle fondamental dans la structuration du discours et est nommé espace de signation. Les personnes nées sourdes, qui ont une expérience du monde basée sur la perception visuelle et non sonore, expriment avoir un mode de pensée visuel qui leur fait privilégier des constructions linguistiques qui permettent de dire mais aussi de montrer.

## 0.2 Contexte général :

Les langues des signes (LS) sont des langues naturelles, visuelles et gestuelles, utilisées par les communautés sourdes pour communiquer. Produites à l'aide des mains, du visage, du regard et du corps, elles s'expriment dans un espace appelé espace de signation. Les personnes sourdes, ayant une perception visuelle du monde, développent un mode de pensée visuel qui influence leur manière de s'exprimer.

Dans un monde technologique en pleine évolution, il devient essentiel de développer des outils pour faciliter la communication entre sourds et entendants, notamment grâce à la reconnaissance automatique de la langue des signes. Ce projet s'inscrit dans cette démarche d'inclusion et d'accessibilité.

Deuxième partie

# Parte II : Implémentation technique

## 0.3 Introduction :

Cette partie technique du projet se concentre sur le développement d'un système innovant visant à assurer une communication fluide et fiable entre les personnes sourdes et les personnes entendantes. L'objectif est de traduire la langue des signes en texte ou en avatar 3D interactif, en veillant à préserver la cohérence contextuelle des gestes traduits. Nous détaillerons ici les différentes étapes d'implémentation, depuis la collecte et le traitement des données jusqu'à la modélisation et l'évaluation du système.

## 0.4 Problématique :

Malgré les avancées technologiques, la barrière de communication entre les personnes sourdes utilisant la langue des signes et les entendants reste importante. Les systèmes actuels de reconnaissance des gestes sont souvent limités en précision, en vocabulaire ou en capacité à comprendre le contexte des signes. La véritable difficulté réside dans la capacité à interpréter correctement une séquence gestuelle dynamique et à la traduire en un message compréhensible, fidèle à l'intention initiale. Il est donc crucial de concevoir un système capable d'identifier avec fiabilité les gestes de la langue des signes et de restituer leur sens sous une forme accessible

## 0.5 Solutions existantes :

Plusieurs projets récents ont tenté de résoudre la problématique de la traduction automatique de la langue des signes en utilisant des outils de vision par ordinateur et d'apprentissage automatique. La majorité de ces approches reposent sur l'utilisation de **MediaPipe**, une bibliothèque développée par Google, qui permet d'extraire en temps réel les points clés (keypoints) des mains, du visage et du corps. Ces points sont ensuite utilisés comme entrée dans différents types de modèles de classification.

Parmi les solutions les plus courantes, on retrouve :

- **LSTM (Long Short-Term Memory)** : utilisé pour traiter des séquences de gestes dans les vidéos, en exploitant la dimension temporelle. Cette approche est pertinente pour la reconnaissance de mots ou de phrases signées.
- **CNN (Convolutional Neural Networks)** : appliqués directement aux images ou aux cartes de keypoints, les CNN permettent d'identifier des signes statiques (lettres, chiffres).
- **Modèles classiques de machine learning** : comme les Random Forest, SVM, ou k-NN, souvent utilisés lorsque les données sont déjà transformées en vecteurs de caractéristiques (comme les keypoints MediaPipe).

Cependant, malgré l'efficacité de ces approches, plusieurs limitations persistent :

— **Manque de précision** pour certains gestes similaires visuellement (ex : lettres “M”, “N” et “O”),
— **Incapacité à comprendre le contexte** ou l’intention du geste dans une séquence plus large,
— **Faible généralisation** lorsque le modèle est testé sur des personnes ou des environnements différents de ceux du jeu de données d’entraînement,
— **Traitement limité à une seule main** dans certains systèmes, ce qui empêche la reconnaissance complète de la langue des signes.

Ces constats motivent le développement d’un système plus robuste, capable de combiner la détection fine des gestes individuels et la reconstruction de leur signification dans un contexte cohérent.

## 0.6 Solutions proposées :

La solution que nous proposons s’inspire des approches existantes, mais cherche à les améliorer sur plusieurs aspects clés. Elle repose sur une architecture modulaire composée de plusieurs étapes, allant de l’extraction des gestes à la reconstruction grammaticale des phrases.

— **Extraction de caractéristiques** : nous utilisons **MediaPipe** pour détecter les keypoints des mains à partir des images ou vidéos. Ces coordonnées représentent la position 3D des différentes articulations de la main, et servent de base pour la reconnaissance des signes.
— **Classification des gestes** : deux types de modèles sont envisagés :
  — Des **modèles classiques** (comme Random Forest ou SVM) pour la reconnaissance des signes statiques (par exemple : les lettres de l’alphabet),
  — Un modèle **LSTM (Long Short-Term Memory)** pour les séquences de gestes, permettant de traiter des mots ou phrases entiers issus de plusieurs frames.
— **Post-traitement linguistique** : une fois les signes traduits en mots, nous intégrons une phase de **traitement automatique du langage naturel (NLP)** afin de corriger les erreurs grammaticales et restructurer les mots reconnus en une phrase cohérente. Cela permet de produire un texte final lisible, fidèle à l’intention du signant.

Cette approche hybride permet ainsi de combiner la puissance de la vision par ordinateur avec l’intelligence linguistique des modèles NLP, en vue d’une traduction automatique plus fluide et naturelle de la langue des signes vers le texte.

### 0.6.1 Prétraitement : Extraction des points clés (keypoints)

Chaque image ou frame vidéo contenant un signe est traitée à l’aide de la bibliothèque **MediaPipe Hands**, qui détecte automatiquement les articulations de la main.

MediaPipe retourne un ensemble de **21 points clés** par main. Chaque point clé est représenté dans un espace tridimensionnel par ses coordonnées $(x, y, z)$. On définit formellement :

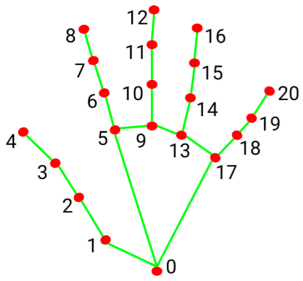
$$\mathbf{K} = \left\{ \mathbf{k}_i = (x_i, y_i, z_i) \in \mathbb{R}^3 \;\middle|\; i = 1, 2, ..., 21 \right\}$$

*Cette formule représente l'ensemble des 21 keypoints détectés par MediaPipe pour une main. Chaque point $\mathbf{k}_i$ est défini par ses coordonnées spatiales dans l'image. Les coordonnées $x$ et $y$ sont normalisées (valeurs entre 0 et 1), et $z$ représente la profondeur relative par rapport à la paume.*

Ces coordonnées sont ensuite **aplaties** dans un vecteur de dimension 63 (car $21 \times 3 = 63$) :

$$\mathbf{v} = [x_1, y_1, z_1, x_2, y_2, z_2, \ldots, x_{21}, y_{21}, z_{21}] \in \mathbb{R}^{63}$$

*Ce vecteur $\mathbf{v}$ regroupe toutes les coordonnées 3D des 21 articulations de la main dans une seule structure linéaire. Il est ensuite utilisé comme vecteur de caractéristiques (features) pour alimenter le modèle de classification.*




**Normalisation :** Avant l'entraînement, chaque vecteur est centré et réduit à l'aide de la transformation suivante :

$$v'_j = \frac{v_j - \mu_j}{\sigma_j}$$

*Cette formule correspond à une standardisation de chaque dimension du vecteur de caractéristiques. Elle permet de centrer les données autour de zéro et de les réduire à un écart-type unitaire, ce qui améliore la stabilité et la performance de l'apprentissage.*

où :
— $v_j$ est la $j$-ème composante du vecteur $\mathbf{v}$,
— $\mu_j$ est la moyenne des valeurs de la composante $j$ sur l'ensemble du dataset,

— $\sigma_j$ est l'écart-type correspondant.

*En d'autres termes, cette étape permet de compenser les variations d'échelle entre les différentes dimensions du vecteur et de garantir que toutes les caractéristiques sont traitées équitablement par le modèle.*

### 0.6.2 Entraînement du modèle :

Après les étapes de prétraitement et de normalisation, les données sont divisées en deux ensembles :

— **Train** : 80% des données, utilisées pour l'entraînement du modèle.

— **Test** : 20% des données, utilisées pour évaluer la performance.

**1. Reconnaissance des alphabets :** Pour la classification des lettres de l'alphabet en langue des signes, nous utilisons deux modèles classiques :

— **Random Forest** :

Le modèle construit $T$ arbres de décision. Pour une entrée $\mathbf{x}$, chaque arbre $t_i$ produit une prédiction $y_i$. La sortie finale est obtenue par vote majoritaire :

$$\hat{y} = \text{mode}(y_1, y_2, \ldots, y_T)$$

— **SVM (Support Vector Machine)** :

Le SVM cherche un hyperplan $\mathbf{w} \cdot \mathbf{x} + b = 0$ qui sépare les classes avec une marge maximale. L'optimisation se fait via :

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{sous la contrainte} \quad y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$$

**2. Reconnaissance des mots via séquences :** Pour la reconnaissance des mots complets à partir de séquences de gestes, nous utilisons un modèle **LSTM (Long Short-Term Memory)**, bien adapté aux données temporelles.

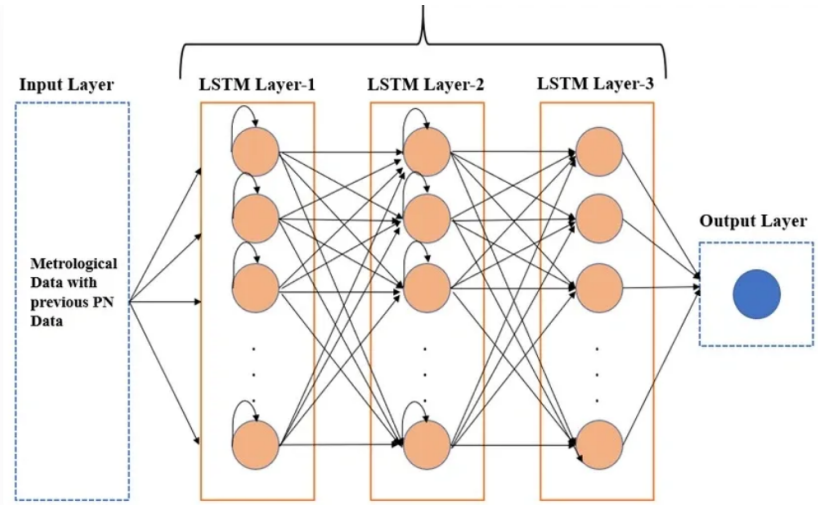
On a une séquence d'entrée $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$, où chaque $\mathbf{x}_t$ est un vecteur des keypoints à l'instant $t$.

Le LSTM est un type de réseau de neurones qui traite ces données dans l'ordre, un pas à la fois. À chaque pas $t$, il reçoit le vecteur $\mathbf{x}_t$ et une information appelée état caché $\mathbf{h}_{t-1}$ provenant du pas précédent.

Le LSTM combine ces deux informations pour calculer un nouvel état caché $\mathbf{h}_t$, qui contient ce qu'il a appris jusqu'à l'instant $t$.

Enfin, après avoir traité toute la séquence, la sortie finale $\mathbf{h}_T$ est utilisée pour prédire la classe de la séquence grâce à une couche dense avec la fonction d'activation softmax :

$$\hat{y} = \text{softmax}(\mathbf{W}\mathbf{h}_T + \mathbf{b})$$




Cette sortie $\hat{y}$ donne la probabilité que la séquence appartienne à chaque catégorie possible.

### 0.6.3 Évaluation du modèle

Après l'entraînement, les performances des modèles sont évaluées à l'aide de métriques classiques en classification : **exactitude (accuracy)**, **précision**, **rappel** et **F1-score**.

Soient les éléments suivants :

— $TP$ : Vrais positifs (True Positives)
— $TN$ : Vrais négatifs (True Negatives)
— $FP$ : Faux positifs (False Positives)
— $FN$ : Faux négatifs (False Negatives)

**1. Exactitude (Accuracy) :** Mesure la proportion de bonnes prédictions parmi l'ensemble des prédictions :

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

**2. Précision (Precision) :** Indique la proportion des exemples prédits comme positifs qui sont réellement positifs :

$$\text{Precision} = \frac{TP}{TP + FP}$$

**3. Rappel (Recall) :** Mesure la proportion des exemples réellement positifs qui ont été correctement identifiés :

$$\text{Recall} = \frac{TP}{TP + FN}$$

**4. F1-score :** Moyenne harmonique entre la précision et le rappel, utile en cas de classes déséquilibrées :

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

**6. Score global (Macro / Micro) :** En classification multi-classes :

- **Macro-averaging** : moyenne non pondérée des métriques pour chaque classe.
- **Micro-averaging** : calcule les métriques globalement sur l'ensemble des instances.

**Performances du modèle(Random forest ) de la reconnaissance des lettres**

**Performances du modèle (SVM) pour la reconnaissance des lettres**

| Classe | Précision | Rappel | F1-score | Support |
|---|---|---|---|---|
| A | 0.98 | 0.99 | 0.99 | 984 |
| B | 1.00 | 1.00 | 1.00 | 1137 |
| C | 0.98 | 0.99 | 0.99 | 854 |
| D | 1.00 | 0.99 | 1.00 | 1131 |
| E | 0.99 | 0.99 | 0.99 | 997 |
| F | 1.00 | 1.00 | 1.00 | 1497 |
| G | 1.00 | 0.99 | 1.00 | 1125 |
| H | 0.99 | 1.00 | 0.99 | 1125 |
| I | 0.99 | 0.99 | 0.99 | 1084 |
| J | 1.00 | 1.00 | 1.00 | 1038 |
| K | 0.99 | 0.99 | 0.99 | 1287 |
| L | 1.00 | 1.00 | 1.00 | 1276 |
| M | 0.96 | 0.99 | 0.97 | 538 |
| N | 0.98 | 0.95 | 0.97 | 473 |
| O | 0.99 | 0.99 | 0.99 | 952 |
| P | 1.00 | 1.00 | 1.00 | 888 |
| Q | 1.00 | 1.00 | 1.00 | 887 |
| R | 0.99 | 0.98 | 0.98 | 1147 |
| S | 0.98 | 0.99 | 0.98 | 1011 |
| T | 1.00 | 0.98 | 0.99 | 1087 |
| U | 0.98 | 0.98 | 0.98 | 1157 |
| V | 0.99 | 0.99 | 0.99 | 1146 |
| W | 1.00 | 1.00 | 1.00 | 1151 |
| X | 0.99 | 1.00 | 0.99 | 1028 |
| Y | 1.00 | 0.99 | 1.00 | 1095 |
| Z | 1.00 | 1.00 | 1.00 | 990 |
| del | 0.99 | 1.00 | 1.00 | 582 |
| space | 0.99 | 0.98 | 0.99 | 499 |
| **Exactitude (accuracy)** | 0.99 (sur 28 166 exemples) | | | |
| **Moyenne macro** | 0.99 | 0.99 | 0.99 | 28 166 |
| **Moyenne pondérée** | 0.99 | 0.99 | 0.99 | 28 166 |

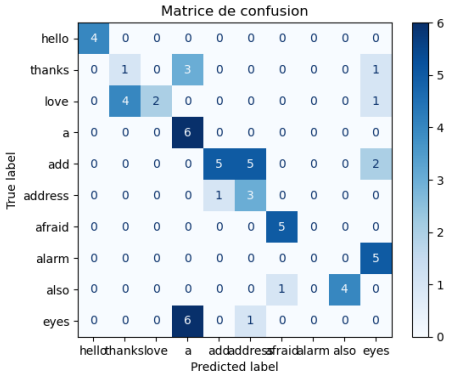
**5. Matrice de confusion :** Pour une classification multi-classes, une matrice de confusion est également utilisée. Elle permet de visualiser les erreurs de classification en affichant pour chaque classe réelle le nombre de fois qu'elle a été

| Classe | Précision | Rappel | F1-score | Support |
|---|---|---|---|---|
| 0 | 0.98 | 0.99 | 0.99 | 984 |
| 1 | 1.00 | 1.00 | 1.00 | 1137 |
| 2 | 0.98 | 0.99 | 0.99 | 854 |
| 3 | 0.99 | 1.00 | 0.99 | 1131 |
| 4 | 0.99 | 0.99 | 0.99 | 997 |
| 5 | 1.00 | 1.00 | 1.00 | 1497 |
| 6 | 0.98 | 0.99 | 0.99 | 1125 |
| 7 | 0.99 | 0.99 | 0.99 | 1125 |
| 8 | 0.98 | 0.97 | 0.98 | 1084 |
| 9 | 0.99 | 0.98 | 0.99 | 1038 |
| 10 | 0.98 | 0.99 | 0.99 | 1287 |
| 11 | 1.00 | 1.00 | 1.00 | 1276 |
| 12 | 0.90 | 0.94 | 0.92 | 538 |
| 13 | 0.95 | 0.89 | 0.92 | 473 |
| 14 | 0.99 | 0.99 | 0.99 | 952 |
| 15 | 0.99 | 0.99 | 0.99 | 888 |
| 16 | 0.99 | 0.99 | 0.99 | 887 |
| 17 | 0.96 | 0.97 | 0.97 | 1147 |
| 18 | 0.99 | 0.99 | 0.99 | 1011 |
| 19 | 0.99 | 0.99 | 0.99 | 1087 |
| 20 | 0.98 | 0.97 | 0.97 | 1157 |
| 21 | 0.99 | 0.97 | 0.98 | 1146 |
| 22 | 0.99 | 0.99 | 0.99 | 1151 |
| 23 | 0.99 | 0.99 | 0.99 | 1028 |
| 24 | 0.99 | 0.99 | 0.99 | 1095 |
| 25 | 0.99 | 0.99 | 0.99 | 990 |
| 26 | 0.98 | 0.99 | 0.99 | 582 |
| 28 | 0.98 | 0.98 | 0.98 | 499 |
| **Accuracy** | | 0.99 | | 28166 |
| **Macro avg** | 0.98 | 0.98 | 0.98 | 28166 |
| **Weighted avg** | 0.99 | 0.99 | 0.99 | 28166 |

TABLE 1 – Rapport de classification des performances du modèle

prédite comme chaque autre classe.

**Martice de confusion Pour la reconnaissance des lettres**




**6. Courbe d'apprentissage :** La courbe d'apprentissage représente l'évolution de la performance du modèle en fonction du nombre d'exemples d'entraînement. Elle permet de diagnostiquer les problèmes de biais ou de variance. On y trace généralement la précision ou l'erreur sur l'ensemble d'apprentissage (training set) et sur l'ensemble de validation/test.

— Une petite différence entre les courbes indique un bon équilibre entre biais et variance.

— Un écart important peut révéler un surapprentissage (overfitting) ou un sous-apprentissage (underfitting).

**Erreur d'apprentissage (training error)** :

$$E_{\text{train}} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}(f(x_i) \neq y_i)$$

**Erreur de validation (validation error)** :

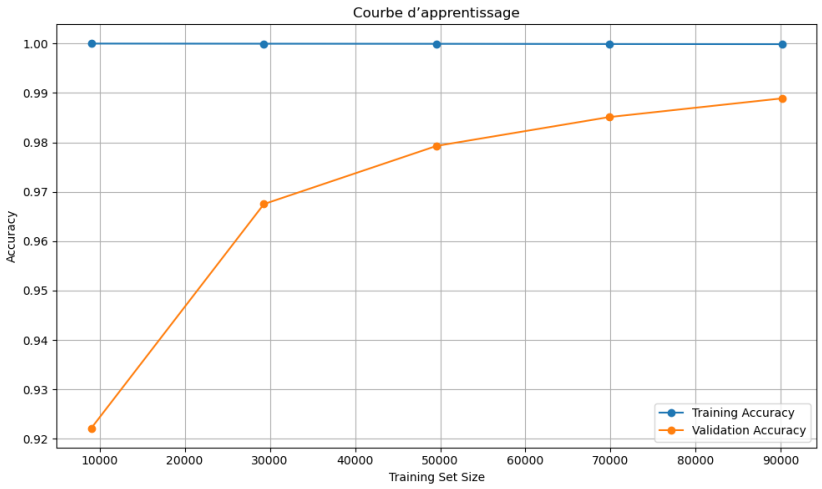
$$E_{\text{val}} = \frac{1}{m} \sum_{j=1}^{m} \mathbb{1}(f(x_j) \neq y_j)$$

où $f(x)$ est la prédiction du modèle, $y$ est la vraie étiquette, et $\mathbb{1}$ est la fonction indicatrice.

**Interprétation typique :**

— Si les deux erreurs sont élevées → **sous-apprentissage** (le modèle est trop simple).

— Si l'erreur sur le test est élevée mais basse sur l'entraînement → **surapprentissage** (le modèle est trop complexe).




*Exemple de courbe d'apprentissage montrant une convergence correcte entre les performances d'entraînement et de validation.*

Ces métriques permettent une évaluation fine de la performance, au-delà de la simple exactitude, en tenant compte de la nature des erreurs (faux positifs et faux négatifs).

## 0.7 Comparaison :

**L'alphabet** : Nous avons testé deux modèles classiques, le SVM et le Random Forest. On observe que les deux modèles obtiennent des précisions assez similaires, ce qui souligne leur pertinence pour la reconnaissance de l'alphabet en langage des signes.

## 0.8 Conclusion

Cette partie technique a démontré la **faisabilité** d'un système de **reconnaissance automatique** de la **langue des signes**, combinant la **vision par ordinateur**, des **modèles de classification** et le **traitement du langage**. Les **résultats obtenus**, notamment avec les modèles **Random Forest** et **SVM**, montrent une **précision élevée** dans la reconnaissance des **lettres**. L'ajout du modèle **LSTM** permet quant à lui d'interpréter des **séquences gestuelles complexes**, ouvrant la voie à une **traduction plus fluide** et **contextuelle**. Ce système constitue ainsi une **base solide** pour améliorer l'**accessibilité** et la **communication** entre **sourds** et **entendants**.

# Troisième partie

# Bibliographie

# Bibliographie


[1] Lugaresi, C., et al. *MediaPipe : A Framework for Building Perception Pipelines*. Google Research, 2019.
Disponible sur : `https://google.github.io/mediapipe/`

[2] Hochreiter, S., Schmidhuber, J. *Long Short-Term Memory*. Neural Computation, 1997.
DOI : `https://doi.org/10.1162/neco.1997.9.8.1735`

[3] Breiman, L. *Random Forests*. Machine Learning, 2001.
DOI : `https://doi.org/10.1023/A:1010933404324`

[4] Cortes, C., Vapnik, V. *Support-vector networks*. Machine Learning, 1995.
DOI : `https://doi.org/10.1007/BF00994018`

[5] Camgoz, N.C., Koller, O., Hadfield, S., Bowden, R.
*Sign Language Transformers : Joint End-to-end Sign Language Recognition and Translation*.
CVPR 2020. Disponible sur :
`https://openaccess.thecvf.com/content_CVPR_2020/papers/Camgoz_Sign_Language_Transformers_`

[6] Kaggle. *American Sign Language Letters Dataset*.
`https://www.kaggle.com/datasets/datamunge/sign-language-mnist`

[7] Joulin, A., Grave, E., Bojanowski, P., Mikolov, T.
*Bag of Tricks for Efficient Text Classification*. arXiv preprint, 2016.
`https://arxiv.org/abs/1607.01759`