جامعة تشرين
كلية الهندسة الميكانيكية والكهربائية
قسم هندسة الحاسبات والتحكم الآلي
السنة الخامسة

تصنيف الجنس بالاعتماد على الصوت باستخدام الشبكات العصبونية الالتفافية CNN

أُعد لنيل إجازة في هندسة الحاسبات والتحكم الآلي

إعداد الطلاب:

أحمد محمود شهلا **أنور فراس ماخوس**

أوس محمد حسن

إشراف:

د. مجد علي

2020-2019

تصادق لجنة الحكم بعد قراءتها ومناقشتها للمشروع، أن المشروع ملائم من حيث النوعية والأهمية ليكون بحثاً لمشروع السنة الخامسة.

We exam committee, certify that we have read this project and that in our opinion it is fully adequate, in scope and quality, as a dissertation for the fifth-year project.

Exam committee:

ملخص:

يهدف هذا المشروع إلى إنشاء نظام تصنيف يمكنه تحديد جنس المتكلم، عن طريق تعليم شبكة عصبونية التفافية وتدريبها على مجموعة بيانات صوتية، بعد استخراج السمات القابلة للتعلم من تلك البيانات الصوتية باستخدام خوارزمية MFCC، قمنا بشرح بعض مفاهيم ضمن هذا التوثيق تتعلق بأساسيات معالجة الإشارة والصوت مثل تحويل فورييه وتحويل تجيب تمام الزاوية المتقطع DCT وغيرها من المبادئ الضرورية لفهم مراحل خوارزمية MFCC الخاصة باستخلاص السمات من المقاطع الصوتية، كما استعرضنا بعض المفاهيم المتعلقة بالذكاء الاصطناعي والشبكات العصبونية وأساسيات طريقة عمل الشبكة العصبونية الالتفافية، واستعرضنا بعض خواص أدوات البحث المستخدمة، من برمجيات وبيانات.

بعد البحث عن بعض نماذج الشبكات العصبونية المستخدمة في دراسات مشابهة، اعتمدنا أحد النماذج لشبكة عصبونية التفافية لتحقيق النظام، واستعملنا مجموعتي بيانات معدتين مسبقاً، الأولى لتدريب الشبكة والثانية للاختبار؛ قمنا ببعض عمليات المعالجة الأولية على الإشارات الصوتية لاستخلاص المعلومات المهمة منها، ثم استخرجنا السمات من الإشارات الصوتية باستخدام خوارزمية MFCC، ومن ثم دربنا الشبكة على مجموعة البيانات الأولى وحددنا دقة التدريب، واختبرنا الشبكة على مجموعة البيانات الثانية وحددنا دقة الاختبار، كما قمنا بإنشاء مجموعة بيانات إضافية خاصة بنا، واختبرنا عليها الشبكة مجدداً لتقييم أداء النموذج في الظروف المختلفة ولتقييم قدرته على التعميم، وحددنا دقة الاختبار الناتجة، ثم نظمنا جدول في نهاية الفصل الخامس لمناقشة النتائج، وفي نهاية بحثنا اقترحنا بعض التطويرات المستقبلية.

Abstract:

This project aims to create a classification system that can determine the gender of the human speaker, by teaching a convolutional neural network (CNN) model and training it on a data set consisted of audio data, after extracting the learnable features from that audio data using the MFCC algorithm. We have explained some concepts within this documentation related to the basics of signal and sound processing such as Fourier transform, DCT, and other principles, which are necessary to understand the stages of the MFCC algorithm for extracting features from audio clips. We also reviewed some concepts related to artificial intelligence, artificial neural networks and the methodology of convolutional neural network, and we reviewed some of the properties of the research tools we have used, including software and data.

After searching for some neural network models, in similar studies, we adopted one of the models for a convolutional neural network to achieve the system. We used two pre-prepared data sets, the first one for training the network and the second one for testing the model. We did some pre-processing operations on the audio signals, to extract the important information from them. Then we extracted the features from the audio signals using the MFCC algorithm. After that, we trained the network on the first dataset and determined the training accuracy, and we tested the network on the second dataset and determined the accuracy of the test. We also created our own data set. Then we also tested the network again to evaluate the model's performance in different conditions and its ability to generalize. We determined the accuracy of the resulting test. Then we organized a table at the end of the fifth chapter to discuss the results, and at the end of our research, we suggested some future developments.

جدول المحتويات

جدول الأشكال

1-الفصل الأول

1-1-ملخص عن فصول الوثيق:

يستعرض الفصل الأول أبرز التطبيقات التي تستخدم أنظمة مشابهة لتصنيف جنس المتكلم وفق الإشارة الصوتية، ويتحدث هذا الفصل عن هدف المشروع والغاية التي يمكن تحقيقها من النظام، والتطبيقات التي يمكن أن تستخدمه.

يتناول الفصل الثاني أساسيات الذكاء الاصطناعي وفروعه من تعلم الآلة والتعلم العميق والشبكات العصبونية وآلية عملها، كما يتناول شرحاً عن الشبكات العصبونية الالتفافية وبنيتها ومكوناتها الأساسية ونستعرض في طيات هذا الفصل آلية عمل الشبكات العصبونية الالتفافية وخوارزمية الانحدار التدريجي المستخدمة في تعليم الشبكات العصبونية.

يتحدث الفصل الثالث عن مبادئ معالجة الإشارة الصوتية والعمليات الأساسية عليها مع المعادلات والتوضيحات اللازمة، ومفاهيم التقطيع إلى نوافذ وإطارات، كذلك يتحدث عن خوارزمية MFCC المستخدمة في استخراج السمات من الإشارة الصوتية، مع شرح خطوات الخوارزمية والتحويلات الرياضية المستخدمة فيها.

يستعرض الفصل الرابع أدوات البحث المستخدمة في مشروعنا، من برمجيات وبيانات، كما يستعرض خدمة Google Colab محرر النصوص البرمجية الذي استخدمناه، ولغة البرمجة المستخدمة Python والمكاتب التي استعملناها، كذلك يتناول ملخصاً عن مجموعات البيانات التي استخدمناها ومواصفات كل منها.

يتناول الفصل الخامس تحقيق النظام ومراحل برمجته، والآليات المستخدمة لقراءة البيانات الصوتية وتحضيرها واستخراج السمات منها، وبناء الشبكة العصبونية الالتفافية وتدريبها، ومن ثم اختبارها وتقييم النتائج.

يستعرض الفصل السادس والأخير بعض المشكلات التي واجهتنا وآلية حلها، مع بعض مقترحات التطوير المستقبلية.

1-2-مقدمة:

تزداد أهمية الحواسيب في حياتنا اليومية، ويصبح التفاعل بين الانسان والآلة ضرورياً أكثر يوما بعد يوم. إن رغبة الإنسان للتواصل مع الآلات بطريقة طبيعية قادت إلى تطور معالجة اللغات الطبيعية. مع التقدم في هذا المجال، أصبح من المحتمل أن يتم استبدال لوحات المفاتيح الحالية بأنظمة تفاعل صوتية في المستقبل. في هذه الأيام إذا أخذنا نظرة في سوق التكنلوجيا حولنا سنجد أن بعض الشركات العالمية

بدأت بالفعل بتطبيق أنظمة تفاعلية مثل نظام Google Assistant لشركة Google ونظام SIRI في شركة Apple وقد لاقت هذه الأنظمة قبولا وأثبتت نجاحها في الأداء، ولتحقيق تقدم دائم في هذا المجال يجب العمل الدائم على تحسين أداء هذه الأنظمة بشكل مستمر ويعد التعرف على جنس المتحدث ضروري جدا في سياق تحقيق هذا التقدم. التعرف على جنس المتحدث من الصوت سواء كان ذكر أم أنثى يستخدم في العديد من المجالات في مجال معالجة اللغات الطبيعية. أظهرت دراسات أن نماذج التعرف على الكلام المعتمة على الجنس أعطت دقة أفضل من النماذج غير المعتمدة على الجنس. أحدث أنظمة التعرف على الكلام التي أطلقتها Google والتي يمكن رؤيتها ضمن أنظمة Android و Google Glass تقوم بالبداية بتحديد جنس المتكلم ثم تقوم بالتعرف على الكلام للبحث. ودقة التعرف على الكلام الناتجة عن هذه الأنظمة تعتبر أفضل بكثير من أنظمة التعرف على الكلام السابقة والتي لم تكن تعتمد على تحديد الجنس.

مؤخرا قامت شركة اعتمادا على نظام Kinect الخاصة بـ بإطلاق غرف ملابس عبر النت التي تقوم بتحديد جنس الشخص المستخدم لها لتقوم باقتراح ملابس مناسبة له.

1-3-الهدف والغاية من المشروع:

نهدف إلى تدريب مجموعة من العينات الصوتية باستخدام شبكة عصبونية التفافية CNN لتصنيف جنس المتحدث من الصوت واختبار الشبكة على مجموعة من عينات الاختبار.

الغاية من المشروع الحصول على منظومة نكون قادرين من خلالها على التعامل مع المقاطع الصوتية المسجلة بصيغة wav وإجراء عمليات معالجة عليها ومن ثم استخلاص السمات المميزة لها باستخدام خوارزمية MFCC واستخدام هذه السمات كدخل للشبكة العصبونية الالتفافية التي قمنا ببنائها وذلك بغية تحديد جنس المتحدث ضمن المقاطع الصوتية فيما إذا كان ذكرا أم أنثى، ويمكن استعمال النظام في تطبيقات مختلفة مثل تصنيف جنس المتكلم في أنظمة التعرف على اللغات الطبيعية، بحيث نصمم نموذجي تعرف على اللغات الطبيعية، الأول يتم تدريبه على سجلات صوتية لذكور والثاني يتم تدريبه على سجلات صوتية لإناث، وعند الاختبار يوضع نظام مشابه لنظامنا في مرحلة سابقة لنماذج التعرف على اللغات الطبيعية، مما يساهم في الحصول على نتائج أفضل، كما يمكن استعمال نظام تصنيف جنس المتكلم في تطبيقات تجارية كمتجر إلكتروني بمساعد صوتي يتعرف على جنس المتكلم ويقدم له اقتراحات مناسبة من المنتجات المتوفرة، وغيرها من التطبيقات العديدة التي يمكن استخدام نظام تصنيفي لجنس المتكلم.

2-الفصل الثاني: الذكاء الاصطناعي وفروعه

2-1-الذكاء الاصطناعي:

الذكاء الاصطناعيّ هو أحد مجالات علوم الحاسب الذي يُشير إلى قدرات المُعالجة المُتقدمة التي تمتلكها الآلات بما يسمح منحها صفة الذكاء، وذلك بالمُقارنة مع الطرق التقليدية التي تستخدمها الآلات والحواسيب في معالجة المُعطيات. مفهوم استخدام الآلات لخوارزميات وطرق معالجة ذكية ليس جديداً بحد ذاته، ولا يمكن اعتبار كل خوارزمية ذكية على أنها ذكاء اصطناعي؛ بل يجب أن تُظهر الآلة قدرةً على تحسين طريقتها الخاصة بحل المشاكل وتوفير الحلول، وذلك عبر التعلم من التُجارب المتتالية التي تمر بها.

هذا الأمر مشابهٌ تماماً للأطفال وطريقة تعلمهم إدراك البيئة المحيطة بهم أثناء سنين النمو، فعند قيام الطفل بالاقتراب من الأشياء الحارة والساخنة لأول مرة لن يدرك أنها ستسبب ألماً له، ولكن بلحظة لمسه لها سيتولد لديه شعور الألم وسيقوم بالابتعاد. في البداية لن يفهم الطفل أن سبب الألم هو الحرارة، وسيعتقد أن الشيء نفسه هو سبب الألم. مع تكرار التجارب سيتعلم الطفل أن سبب الألم هو الحرارة وليس الشيء نفسه، وبالتالي يمكن لأي شيء أن يكون ساخناً أو بارداً أو حتى فاتراً. يمكن فهم هذه التجارب على أنها بيانات يقوم دماغ الطفل بمعالجتها، وفي كل مرة يقوم فيها الطفل بلمس شيءٍ ما، سيتم توليد بيانات جديدة تضاف للبيانات القديمة التي يعلمها الدماغ أصلاً، ومن ثم سيتم معالجة هذه البيانات لينتج عنها معرفة جديدة، وهذا الأمر يتكرر في كل مرة يقوم الطفل بلمس أي شيءٍ من حوله ذي درجة حرارة مختلفة، الطريقة التي تعلم بها الطفل التفريق بين الحرارة نفسها كمفهوم وبين اكتساب الأشياء من حوله لدرجات حرارة مختلفة هي ما يريد الباحثون والعاملون في مجال الذكاء الاصطناعيّ تحقيقه، وهو بناء طرقٍ وآليات تتيح للآلات والحواسيب من حولنا أن تتعلم بشكلٍ مستمر من البيانات المختلفة التي ترد إليها، بما يساعد على تحسين أداء الحواسيب في أداء مهامٍ مختلفة ومتنوعة.

من المهم هنا الانتباه لنقطةٍ هامة، قد يُفهم بشكلٍ خاطئ أن هدف الذكاء الاصطناعيّ هو تزويد الآلات بقدراتٍ ذكية عبر تقليد طريقة عمل الدماغ والعصبونات البشرية، وهو أمرٌ غير دقيق بالمعنى الكامل، إذ إنه وبالرغم من وجود طرق تعتمد على محاكاة آلية عمل العصبونات مثل الشبكات العصبونية الاصطناعية Artificial Neural Networks، إلا أنها ليست النموذج الوحيد المستخدم في الذكاء الاصطناعي، بل يوجد مناهج وطرق وآليات أخرى تستخدم من أجل تزويد الآلات بقدرات معالجة المعلومات وتحليلها وإدراك البيئة المحيطة والتعلم المستمر عبر الخبرة. بهذه الصورة، وعندما يتم وصف تطبيق أو برنامج أو حتى روبوت على أنه معتمد على تقنيات الذكاء الاصطناعيّ، فإن ما يجب معرفته بهذا السياق أن هذا التطبيق (أو البرنامج أو الروبوت) يمتلك القدرة على تحسين نتائج عمله بشكلٍ متواصل اعتماداً على البيانات والمعلومات

التي يحصل عليها، وهذا الأمر يتم عن طريق خوارزمياتٍ وطرقٍ متنوعة. تحديات وجوانب الذكاء الاصطناعي:

لا يمكن النظر اليوم للذكاء الاصطناعيّ على أنه جانبٌ علميّ مُكتمل، فهنالك العديد من الأمور التي لا تزال جدلية ومبهمة بين الباحثين، وذلك بدءاً من سؤالٍ بسيط: ما هو الذكاء؟ عند محاولة الإجابة على هذا السؤال من منظورٍ بشريّ، فإن الذكاء نفسه مفهومٌ ذو إشكالياتٍ كبيرة، هل الذكاء هو القدرة على حل مسائل الرياضيات الصعبة؟ هل هو القدرة على تأليف مقطوعاتٍ موسيقية مبتكرة؟ هل هو القدرة على رسم لوحةٍ فنية تعبيرية؟ هل هو القدرة على التواصل الاجتماعيّ الفعال؟ هل هو كل ما سبق؟ المقصد هنا أن تعريف الذكاء نفسه ليس بالأمر السهل بالنسبة للبشر أنفسهم، وبالتالي فإن إطلاق صفة "الذكاء" على الآلة سيكون أيضاً محط جدلٍ بين الباحثين.

حاول بعض العلماء دراسة هذه المشكلة استناداً للمنطق البشريّ نفسه، أي وضع تعريفٍ أو مفهوم موحد ومتفق عليه للذكاء ومحاولة إسقاطه على الآلة من أجل توصيف الخطوط العريضة والأساسية للذكاء الاصطناعيّ. يعتبر الفيلسوف جاك كوبلاند Jack Copeland من الباحثين البارزين بهذا المجال، والذي حدد العوامل الأساسية للذكاء كما يلي:

1. التعلم العام Generalization Learning أو الذكاء العام General Intelligence: قدرة المتعلم على التأدية بشكلٍ جيد عند وضعه أمام مشاكل لم يتعرض لها من قبل. بحالة الذكاء الاصطناعيّ، فإن هذا يعني القدرة على بناء خوارزمية قابلة للاستخدام في العديد من المجالات والتطبيقات، وليس فقط في مجال واحد ولهدفٍ محدد. لو عدنا لمثال المساعدات الرقمية الذكية (مثل مساعد جوجل أو سيري أو أليكسا) فإنها تمثل نماذج على برمجيات ذكاء اصطناعيّ ذات هدفٍ محدد ولا يمكن استخدامها مثلا من أجل تشخيص الأورام في الصور الإشعاعية الملتقطة باستخدام أجهزة التصوير الطبقي المحوري. الذكاء العام يعني قدرة الخوارزمية على التعلم وإيجاد حلولٍ لمشاكل غير محصورة بمجال محدد، ويعتبر برنامج ألفا جو من جوجل أحد برامج الذكاء الاصطناعيّ ذات الذكاء العام.
2. المنطق Reasoning: أي قدرة المتعلم على التوصل لاستنتاجاتٍ تتناسب مع الموقف أو المشكلة قيد الحل.
3. حل المشاكل Problem Solving والتخطيط للحلول Planning: القدرة على استخدام المعطيات المتوفرة من أجل تطوير حلٍ لمشكلةٍ ما وكيفية تنفيذ هذا الحل.
4. الإدراك Perception: القدرة على تحليل البيئة المحيطة وفهمها وكذلك فهم العلاقة بين الأشياء المختلفة المتواجدة ضمنها.

- فهم اللغات Understanding Languages، أو المعالجة الطبيعية للغة Natural Language Processing وهو ما يندرج ضمنه مشروعنا.

الذكاء الاصطناعي مجال واسع وعميق ومع تطور التقنيات وازدياد الحاجة إلى أنظمة أكثر تطورا ظهرت مصطلح تعلم الآلة Machine learning والذي يعتمد على شبكات عصبونية في عمله ومن ثم ظهر مصطلح التعلم العميق Deep learning الذي يندرج مشروعنا ضمن إطاره.

2-1-1-تعلم الآلة:

يُعتبر تعلم الآلة إحدى الطرق التي تُدرب فيها البرمجية على التعلم والتطور تلقائيًا عن طريق التجربة، حتى دون أن تُبرمَج أو يُغيّر في نصوصها البرمجية من قِبل المبرمجين بشكل صريح، إذ تُستخدَم الخوارزميات والنماذج الإحصائية للتعلم دون أي تدخل بشري، لكن بالرغم من ذلك، يلزم لتعلم الآلة وجود عينة من البيانات المستخدمة في التدريب (بيانات التدريب)، أي البيانات المستخدمة من قِبل الخوارزميات لتوليد نماذج رياضية.

تُصنف مهام تعلم الآلة ضمن عدة فئات وعناوين عريضة:

- التعلم المُراقب أو الخاضع للإشراف Supervised learning: أن تكون البيانات المدخلة مسماة وموسومة بشكل جيد مع المخرجات التي نريدها. كأن تكون البيانات المدخلة مجموعة صور لِقطة ونريد التعرف عليها؛ تُوفَّر مجموعة أخرى من البيانات لاستخدام خوارزمية مُراقبة لإنتاج مخرجات صحيحة (التعرف على القطة) عن طريق البيانات الموسومة.
- التعلم غير المراقب أو غير الخاضع للإشراف Unsupervised learning: عندما تُقدَم البيانات إلى الآلة دون أي مُسمى ويُسمح للخوارزميات بالعمل وفقًا لتلك البيانات ومن دون إرشاد، عندها تجمع الآلة البيانات غير المُصنفة تبعًا لأنماطها وأوجه تشابهها واختلافها دون أي تدريب سابق من مصادر أخرى.
- التعلم المعزز Reinforcement learning: يُستخدم بغية إيجاد السلوك أو الطريق الأفضل والممكن للتعامل مع مشكلة ما، وأفضل مثال على ذلك هو لعبة الشطرنج. عندما نزود البرمجية بالمدخلات الأولية (قواعد اللعبة وأماكن أحجار الشطرنج)، ثم تُدرب على أكبر عدد محتمل من المخرجات كالحركات وإيجاد الحلول للوصول في نهاية المطاف إلى (كش ملك).

2-1-2-التعلم العميق:

يُعتبر أحد فروع تعلم الآلة، إذ تكون الخوارزميات مستوحاة من بنية ووظيفة الدماغ والتي يُطلق عليها تسمية الشبكات العصبونية الاصطناعية Artificial Neural Network، يمكن أن تكون عملية التعلم خاضعة للإشراف أو شبه خاضعة له أو حتى غير مُشرف عليها.

بالإضافة إلى ذلك فإن دقة التعرف على الأشياء (الكلام أو الصور) في التعلم العميق أفضل إلى حد كبير من أي وقت مضى، وذلك لسببين رئيسين:

1. توفر كمية كبيرة من البيانات المُسماة والمُصنفة.
2. الزيادة الكبيرة في قوة الحواسيب بسبب إدخال وحدة معالجة الرسوميات عالية الأداء والحوسبة السحابية.

للتعلم العميق تطبيقات محتملة في العديد من الصناعات، فقد دخل استخدامه في التحكم بالسيارات ذاتية القيادة، وفي المجال الطبي بالكشف عن الخلايا السرطانية بالأشعة السينية واختبارات التصوير الشعاعي، والمساعدة في فعالية الطاقة وكفاءتها، والكثير من الاستخدامات.

وكما ذكرنا سابقًا، يُشار إلى أساليب التعلم العميق على أنها شبكات عصبونية عميقة، إذ يستخدم معظمها شبكات عصبونية اصطناعية. وفي الظروف القياسية لتعلم الآلة، إذا كان هدفنا هو التعرف على الكلب، يجب إدخال مزايا المخرجات المرجوة يدويًا، مثل الكشف عن الكائنات والحواف وما إلى ذلك.

لا تحتاج أساليب التعلم العميق استخراجًا للمزايا اليدوية، فهي مُدربة باستخدام مجموعات كبيرة من البيانات المُسماة والمُصنفة، وهياكل الشبكات العصبونية التي تتعلم هذه المزايا مباشرة عن طريق البيانات.

عادةً ما يُشير مُصطلح "عميق" إلى طبقات مخفية في الشبكة العصبونية، وقد يصل عددها إلى 150، بينما يُقابلها 2–3 فقط في الشبكات العصبونية الاصطناعية، وللحصول على المخرجات المرجوة؛ يستخدم التعلم العميق الكثير من البيانات، ومن الجدير بالذكر أنه من أجل الانتقال عبر تلك الطبقات، يجب تحميل برنامج تشغيل الآلة التي تستخدم التعلم العميق بقدرة حوسبة عالية، وإلا فإن تلك العملية ستستغرق وقتًا أطول بكثير.

2-2-الشبكات العصبونية الاصطناعية:

بما أننا تحدثنا عن تعلم الآلة والتعلم العميق يجب أن نشرح مفهوم الشبكات العصبونية حيث في السنوات الأخيرة، تزايد استخدام مُصطلح الشبكات العصبونية الاصطناعية أو الشبكات العصبية الاصطناعية (Artificial Neural Networks) عند الحديث عن تطبيقات الذكاء الاصطناعيّ في شتى المجالات، إلى الدرجة التي أصبحت فيها الشبكات العصبونية تعني الذكاء الاصطناعيّ عند الكثيرين، وهو أمرٌ غير دقيق، إذ إن الشبكات العصبونية تندرج فعلياً تحت مصطلح التعلم العميق الذي يعتبر أحد أشكال التعلم الآلي، وهو بدوره أحد فروع الذكاء الاصطناعي.

الشبكات العصبونية الاصطناعية عبارة عن برامج أو أنظمة حاسوبية تعتمد من حيث المبدأ على محاكاة عمل عصبونات الدماغ من أجل معالجة البيانات وإنجاز مهامٍ في مجالاتٍ متنوعة، وهي أشهر أنماط وطرق التعلم الآلي الهادف لتوفير خوارزمياتٍ وبرمجياتٍ قادرة على التعلم بالخبرة. عندما يتم القول إن الشبكات العصبونية تحاكي آلية عمل العصبونات الحيوية في الدماغ فهذا يعني أمرين: الأول بنيوي معني بتشكيل الشبكة العصبونية لتتكون من عددٍ معينٍ من العقد تُدعى كل منها "عصبون" مرتبطة مع بعضها البعض عبر وصلاتٍ اصطناعية. والأمر الثاني هو الناحية السلوكية؛ أي أن العصبونات الاصطناعية

تقلد العصبونات الحيوية في كيفية توليدها للإشارات ونقلها فيما بينها، كما أن الشبكة العصبونية ككل يجب أن تحاكي عمل الدماغ من حيث قدرته على التعلم من الخبرة، وكلما ازدادت الخبرة المكتسبة ازدادت قوة المشابك بين العصبونات الحيوية، وبالتالي يصير أداؤنا في تنفيذ مهمةٍ ما أفضل. هذا الأمر نفسه يجب أن ينطبق على الشبكات العصبونية الاصطناعية.

من حيث الشكل العام، تأخذ الشبكة العصبونية شكل طبقاتٍ متعددة تمتلك كل طبقة منها عدداً معيناً من العصبونات، وأقل عدد ممكن من الطبقات هو ثلاث: طبقة الدخل وطبقة الخرج والطبقة (أو الطبقات) المخفية. تقوم طبقة الدخل باستقبال البيانات التي يجب معالجتها، وبحسب قيم بيانات الدخل سيتم تفعيل عصبونات معينة ضمن طبقة الدخل، التي بدورها ستقوم بتفعيل عصبونات الطبقة المخفية، وهكذا وصولاً حتى طبقة الخرج التي نحصل من خلالها على القرار المتعلق بالمهمة المراد إنجازها.

تتألف الشبكة العصبونية من عصبونات مرتبة ضمن طبقات، بحيث يتصل كل عصبون في كل طبقة مع عصبونات الطبقة السابقة والتالية.

2-2-1-آلية عمل الشبكات العصبونية الاصطناعية:

الشبكة العصبونية عبارة عن شبكة من الحسابات والتوابع الرياضية، وذلك بدءاً من الوحدة الأساسية التي تشكل الشبكة، أي العصبون، الذي يتم تشكيله فعلياً عبر تابعٍ رياضيّ يقوم بحساب قيمةٍ معينة.

يمكن تنفيذ ذلك عبر تشكيل شبكةٍ عصبونية تتكون من مجموعة طبقات، طبقة دخل تقوم باستقبال بيانات الدخل، وعدد عصبونات هذه الطبقة سيساوي عدد عناصر مصفوفة الدخل، فإذا كان لدينا الدخل يتضمن 784 بيكسل (أي 28 × 28) فهذا يعني أن طبقة الدخل ستمتلك 784 عصبون، ومجموعة من الطبقات المخفية تقوم بالتعرف على الأنماط الدقيقة، والثانية تقوم بالتعرف على الأنماط الكبيرة التي يمكن تشكيلها من الأنماط الدقيقة المكتشفة في الطبقة المخفية الأولى.

ما الذي يحدث ضمن هذه الشبكة؟ لنعد قليلاً للعصبونات، العصبون نفسه عبارة عن تابع رياضي يقوم بحساب قيمة تدعى التفعيل (Activation)، وهذا يعني أن كل عصبون يمتلك قيمة تفعيل خاصة به، وحساب هذه القيمة يعتمد على قيم تفعيل العصبونات المرتبطة به وعلى قوة تشابكه معها. كيف نحسب "قوة" التشابك؟ هذا الأمر يقودنا لمفهومٍ آخر، وهو أن كل وصلة بين عصبونين تمتلك قيمة عددية تعرف باسم الوزن (Weight).

الآن نريد الانتقال إلى الطبقة المخفية الأولى، ومن أجل حساب قيمة تفعيل العصبون الأول فيها، يجب أن نقوم بضرب قيمة تفعيل كل عصبون من طبقة الدخل بوزن وصلته مع العصبون الأول من الطبقة المخفية، ومن ثم يتم جمعها مع بعضها البعض، ثم تُكرَّر هذه العملية من أجل كافة عصبونات الطبقة المخفية الأولى. بعد حساب قيم تفعيل عصبونات الطبقة المخفية الأولى يتم الانتقال للطبقة المخفية الثانية بنفس الطريقة، وهكذا حتى الوصول لطبقة الخرج

2-2-2-آلية تحديد قيم أوزان الوصلات:

تطرح الشرح السابق لكيفية قيام الشبكة بتنفيذ خطوات المعالجة الخاصة بها، ولكن هناك سؤال هام، طالما تلعب أوزان الوصلات بين العصبونات دوراً هاماً في العمليات الحسابية الخاصة بحساب قيم تفعيل عصبونات الطبقة التالية، لكن كيف يتم تحديد قيم هذه الأوزان؟ هذا ما يقود إلى مفهوم التعلم.

عندما تبدأ الشبكة بالعمل للمرة الأولى فإنها ستمتلك قيماً أولية لأوزان الوصلات (وكذلك بعض البارامترات الأخرى، ولكن سنجعل الشرح هنا مقتصراً على الأوزان للتبسيط)، ومن ثم يتم تمرير مجموعة من عينات الدخل، ونظراً لكون قيم الأوزان غير مضبوطة بشكلٍ دقيق، لن تكون النتائج التي ستقدمها الشبكة صحيحة.

لحسن الحظ، يمكن تزويد الشبكة بآلية تخبرها إن كانت تعمل بشكلٍ صحيح أم لا، وكذلك مدى دقة القرارات التي تخلص إليها. هذا الأمر يتم عبر حساب نسبة التعرف الصحيح التي تقوم به الشبكة مع كل مرة يتم إدخال عينة تدريب إليها، وتزويد هذه المعلومات للشبكة لتقوم بتعديل قيم أوزان الوصلات بين العصبونات. مع تكرار عمليات إدخال عينات التدريب التي ستؤدي إلى تعديل قيم أوزان وصلات العصبونات، ستستطيع الشبكة معرفة الأوزان التي تؤدي لأعلى احتمالية تعرّف صحيح وتلك التي تؤدي لاحتمالية تعرّف خاطئ. هذا الأمر يقود لاستنتاج هام: الشبكات العصبونية تمتلك شهية عالية جداً للبيانات! بمعنى أنه كلما ازدادت كمية بيانات التدريب تحسنت دقة عمل الشبكة.

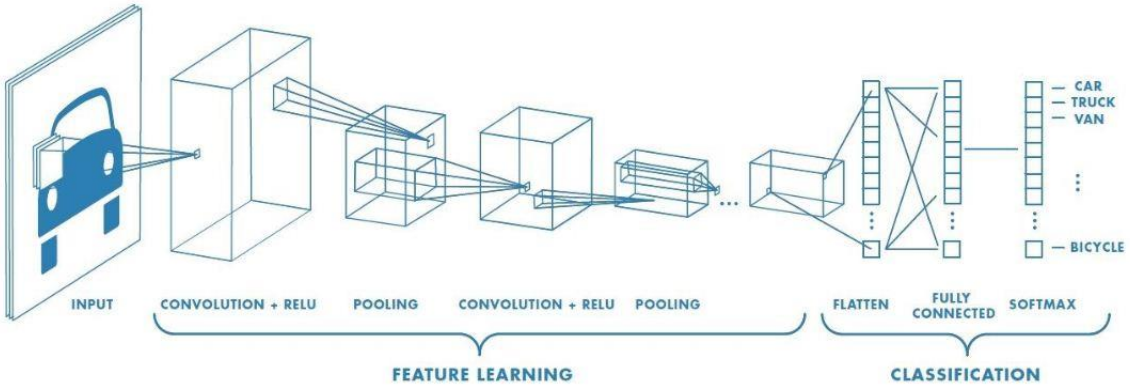
هذا الإجراء السابق هو ما يعرف باسم التعليم (Learning)، والطريقة الموصوفة سابقاً هي ما تعرف باسم: التعلم الموجه (Supervised Learning) أو التعليم الخاضع للإشراف؛ وذلك لأن المبرمج هو من يقوم بإخبار الشبكة العصبونية إن كان عملها صحيحاً أو خاطئاً، وكذلك كيفية تحسين دقة عملها ونتائجها. وهنالك نمط آخر من التعليم لا يتدخل فيه المبرمج ولا يوجد عبره طريقة لإخبار الشبكة عن صحة نتائجها، بل تترك وحدها لتستنتج القرارات المتعلقة بالخرج بناءً على البيانات المدخلة إليها أثناء مرحلة التدريب، وهذا النمط من التعليم يعرف باسم: التعلم غير الموجه (Unsupervised Learning)، أو التعلم غير الخاضع للإشراف.

2-3-الشبكات العصبونية الالتفافية:

2-3-1-مقدمة:

الشبكات العصبونية الالتفافية هي نوع خاص من الشبكات العصبونية بالتغذية الأمامية Feed forward neural network وتستمد إلهامها من العمليات البيولوجية الحاصلة في الفص البصري بالتحديد في دماغ الكائنات الحية، وتعتبر حلا للكثير من مشاكل الرؤية الحاسوبية في الذكاء الاصطناعي.

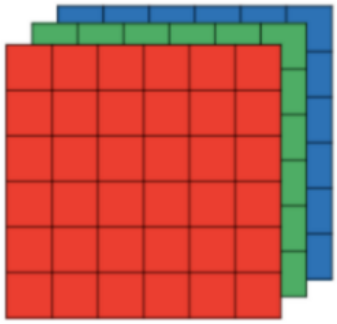
تتكون الشبكات العصبونية الالتفافية من مجموعة من الطبقات التي يمكن تجميعها حسب وظائفها، يبين الشكل (2-1) مثال عن بنية الشبكة العصبونية الالتفافية.



الشكل 2-1 بنية الشبكة العصبونية الالتفافية

2-3-2-دخل الشبكات العصبونية الالتفافية:

لفهم استجابة الشبكات العصبونية الالتفافية وطريقة معالجتها للبيانات المدخلة سوف نعتمد شرح طريقة معالجة الشبكة للمدخلات في مجال معالجة الصورة كون دخل الصور أسهل للمقاربة والتمثيل، وينطبق على استجابة الشبكة لدخل الإشارة الصوتية ما ينطبق على دخل الصورة باختلاف أبعاد الدخل، من الضروري جدا فهم شكل عينات الدريب (بيانات الدخل)، فالصورة مثلا بالنسبة للحاسوب عبارة عن مصفوفة ثلاثية الأبعاد (العرض × الارتفاع × العمق) من قيم تتراوح بين 0-255، تمثل عناصر المصفوفة نقاط لونية (Pixels) متوزعة طولا وعرضا وثلاث قنوات لونية (إذا كان النظام اللوني هو RGB) كما يبين الشكل (2-2):



الشكل 2-2 مصفوفة صورة

مع العلم أنه لو كان عدد القنوات 1 ستكون الصورة grayscale أي بالأبيض والأسود.

2-3-3-المكونات الأساسية للشبكات العصبونية الالتفافية:

المكونات الأساسية للشبكات العصبونية الالتفافية هي:

- الطبقة الالتفافية Convolutional Layer
- طبقة التجميع Pooling Layer

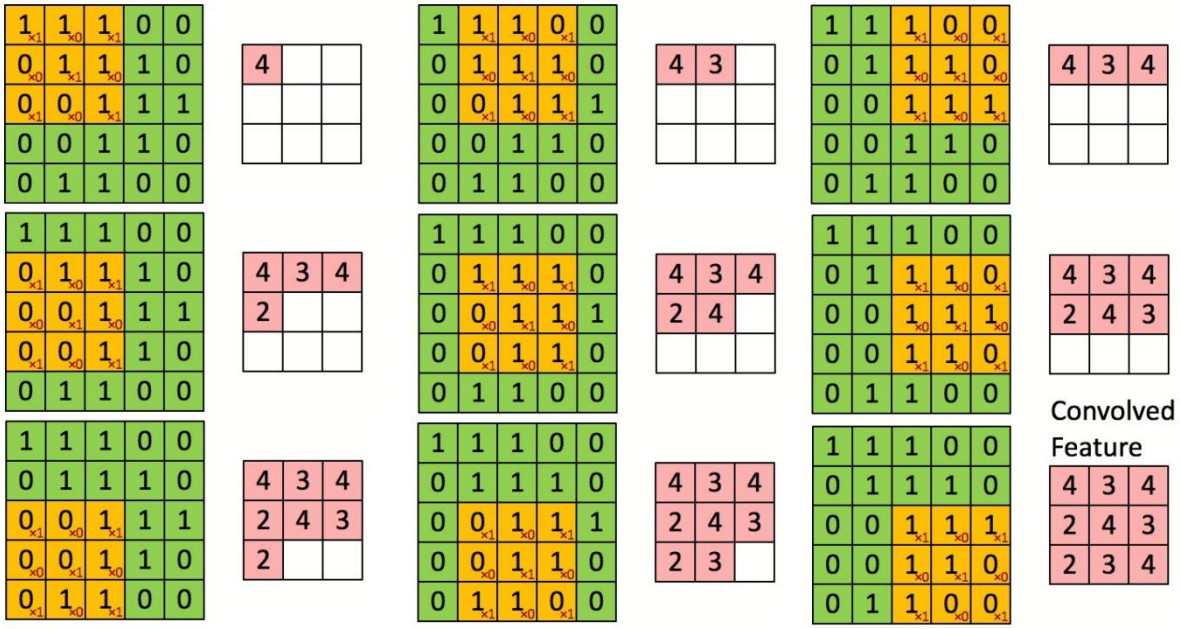
- طبقة الارتباط الكامل Fully-connected Layer

سنوضح كل منها فيما يلي.

2-3-3-1-الطبقة الالتفافية (Convolution Layer):

تحوي هذه الطبقة مرشحا Filter ويعرف أيضا باسم Kernel من شأنه تحديد وجود سمات أو أنماط معينة في الدخل، ويمكن استخدام عدة مرشحات بغية استخراج سمات مختلفة، يكون المرشح ذو حجم صغير ليمسح مصفوفة الدخل كاملة ويطبق العمليات الحسابية المناسبة بين قيم المرشح وعناصر المصفوفة (النقاط اللونية) بغية استخراج السمات (Features) وعادة ما تكون الجداء الداخلي فيما بينهما.

يوضح الشكل (2-3) العمليات الالتفافية.

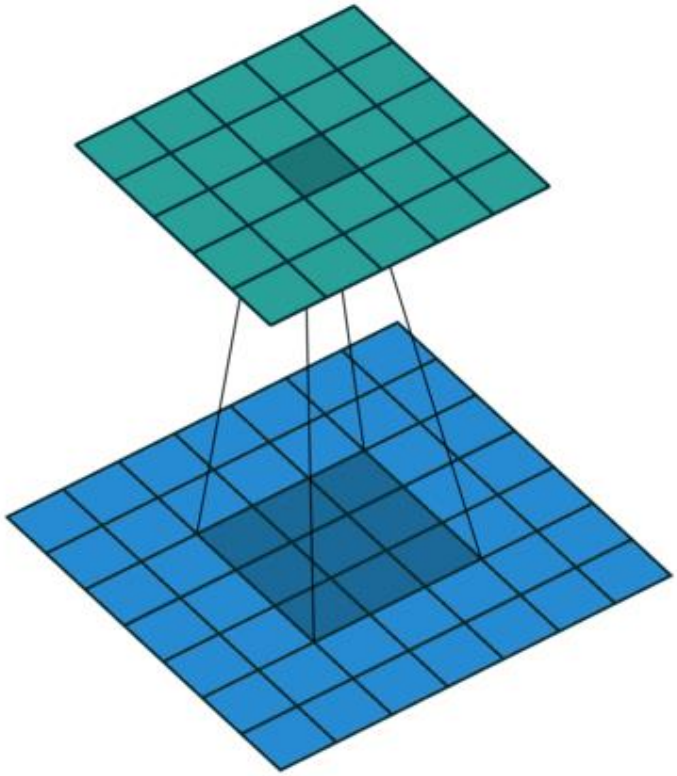


الشكل 2-3 تطبيق العملية الالتفافية على المصفوفة

يعاد ضبط قيم المرشح خلال عملية التدريب الدورية وعند تدريب الشبكة لعدد معين من التكرارات (epochs)، وكل تكرار تدريب يعني إدخال كل أمثلة التدريب مرة واحدة، تشرع هذه المرشحات البحث عن سمات متمايزة في الصورة.

توظف الطبقات المخفية الأولى في استخراج السمات البسيطة والواضحة مثل الحواف في الاتجاهات المُختلفة، وما إلى ذلك. ومع التعمّق أكثر في الطبقات المخفية في الشبكة، تزداد درجة تعقيد السمات التي يجب تحديدها واستخراجها.

نحتاجُ تخزينَ معلوماتٍ عن المواقع المحتوية على شكل المُرشِّح وذلك بتخزين قيم الجداء الداخلي من أجل كل receptive field في مصفوفة أُخرى تُدعى خريطة التفعيل (activation map) أو خريطة السمات (Feature map)؛ اسمها خريطة لأنّها تُعطي معلومات عن وجود المُرشِّح في الصورة ومواقع ظهوره فيها. ويكون عمق الخريطة من أجل كُل مُرشِّح مساويًا إلى (1) كما يبين الشكل (2-4).



الشكل 2-4 تخزين نتائج الجداء الداخلي في Activation map

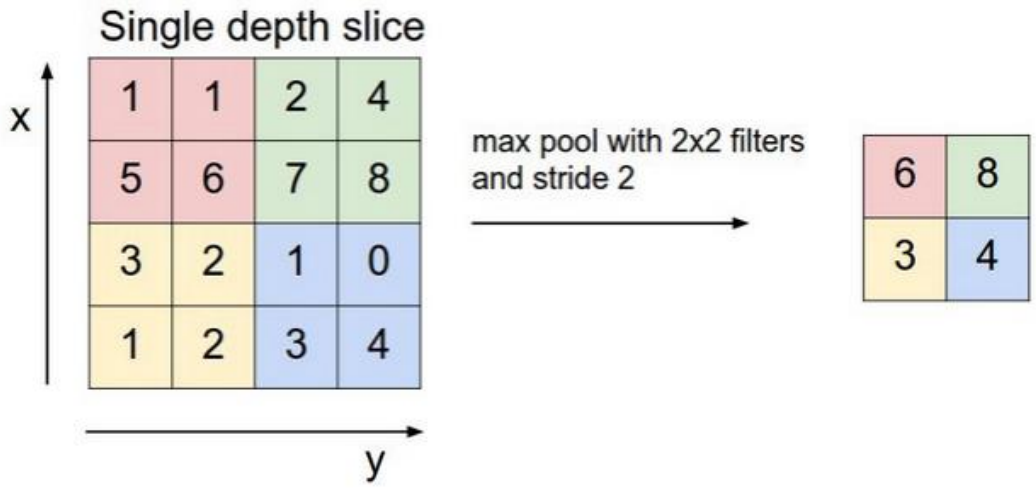
2-3-3-2-طبقة التجميع:

إن الغاية من طبقة التجميع هي تقليص حجم خرائط التفعيل (قلنا خرائط لإمكانية استخدام أكثر من مُرشِّح). لا يُقلل ذلك من كمية الحسابات الضرورية فحسب، بل يقي من الوقوع في حالة overfitting.

الفكرة الكامنة وراء التجميع بسيطة جدًا: تقليص حجم المصفوفات الكبيرة، وتتم العملية من خلال تطبيق إحدى الدالتين الآتيتين:

- Max: تحديد القيم العُظمى في كل نافذة.
- Average: حساب المتوسط الحسابي للقيم الموجودة في النافذة الواحدة.

إن التقنية الأكثر رواجًا هي الأولى Max-pooling: مفادُ هذه الطريقة هو مسح خريطة التفعيل (أو مصفوفة السمات) بنافذة صغيرة والإبقاء على القيم الأكبر ضمن كل نافذة مُقلّصين بذلك حجم الخريطة، كما يبين الشكل (2-5).



الشكل 2-5 تطبيق Max-pooling على المصفوفة

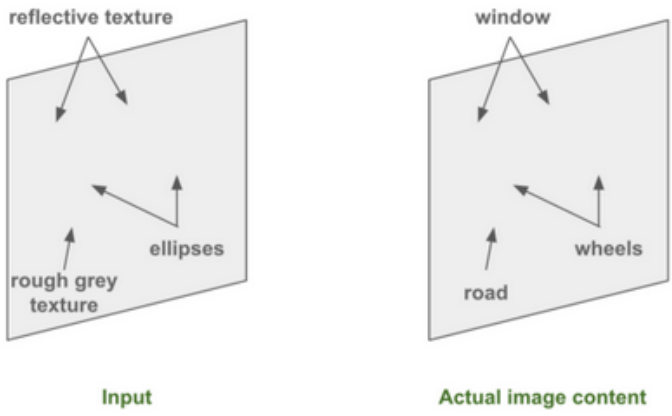
إذن يتم في طبقات التجميع استخراج أفضل القيم للسمات الموجودة في مصفوفة السمات التي نتجت عن طبقة الالتفاف.

2-3-3-3-طبقة الارتباط الكامل:

تكون هذه الطبقة هي الأخيرة في الشبكة العصبونية الالتفافية وهي من نوع (multi-layer perceptron)، حيث العصبونات مُرتبطةً بالكامل مع كُل عُقد الطبقة السابقة لها. سبب وجودها في النهاية لأن عملية التصنيف النهائية تتم فيها.

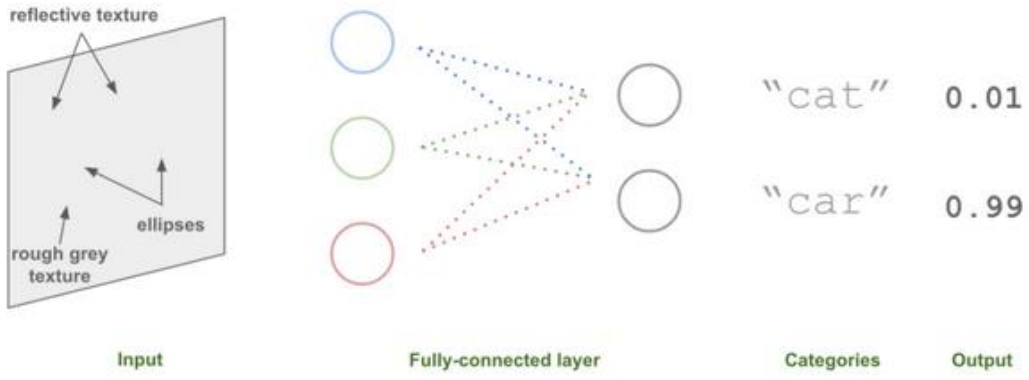
تُطبق فيها عملية بَسْط الدخل (flattening operation) إلى شُعاع سمات ومن ثَم تمريره إلى شبكة من العصبونات للتنبؤ باحتمالات الخرج.

دخل الطبقة هو مصفوفة تتضمن معلومات عن مواقع توضّع أشكال (أنماط) معقدة مُعينة في الصورة، ونُطلق على هذه الأنماط المُعقدة اسم (مُصنِّفات قابلة للتدريب أو trainable classifiers) هي المُرشِّحات ذاتها أو (Kernels) أو مصفوفة الأوزان، يبين الشكل (2-6) الفرق بين الدخل والأنماط التي يتم التعامل معها كدخل.



الشكل 2-6 دخل طبقة Fully-connected Layer

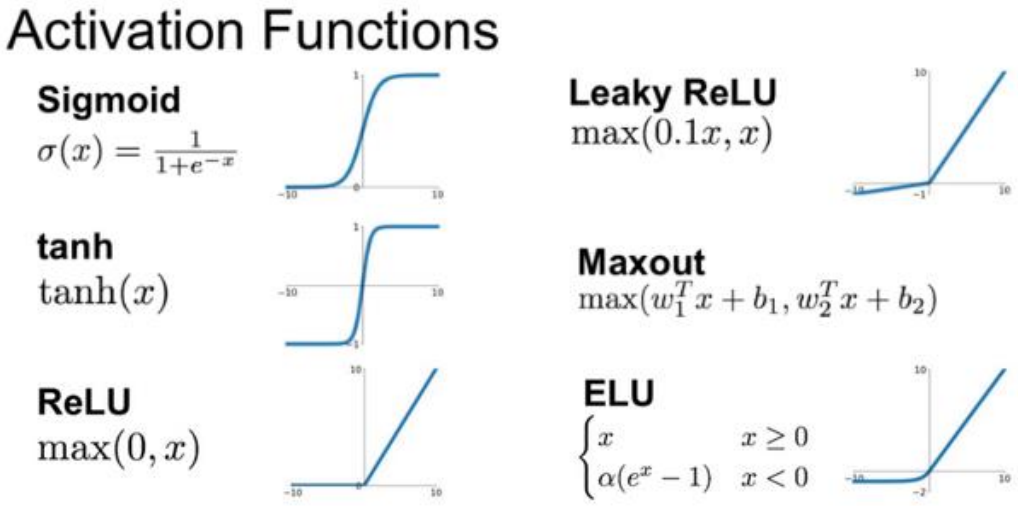
خرج الطبقة هو شعاعُ قيمٍ تمثِّل كل منها احتمال صنف من التصنيفات التي تُدرَّب الشبكة عليها (أي الأوزان)، فمثلاً: بفرض لدينا كل من التصنيفين "cat" و"car"، سيكون خرج الطبقة [0.91، 0.01]، هذا يعني احتمال أن تكون الصورة لِقطة هو 0.91، أما احتمال كونها لِسيارة هو 0.01، وباختيار الاحتمال الأكبر تكون الصورة من صنف "cat" ويوضح الشكل (2-7) هذه العملية.



الشكل 2-7 خرج طبقة Fully-connected Layer

2-3-3-4-دالة التفعيل:

هي دالة لا خطيّة ولها عدّة أنواع مبينة في الشكل (2-8).



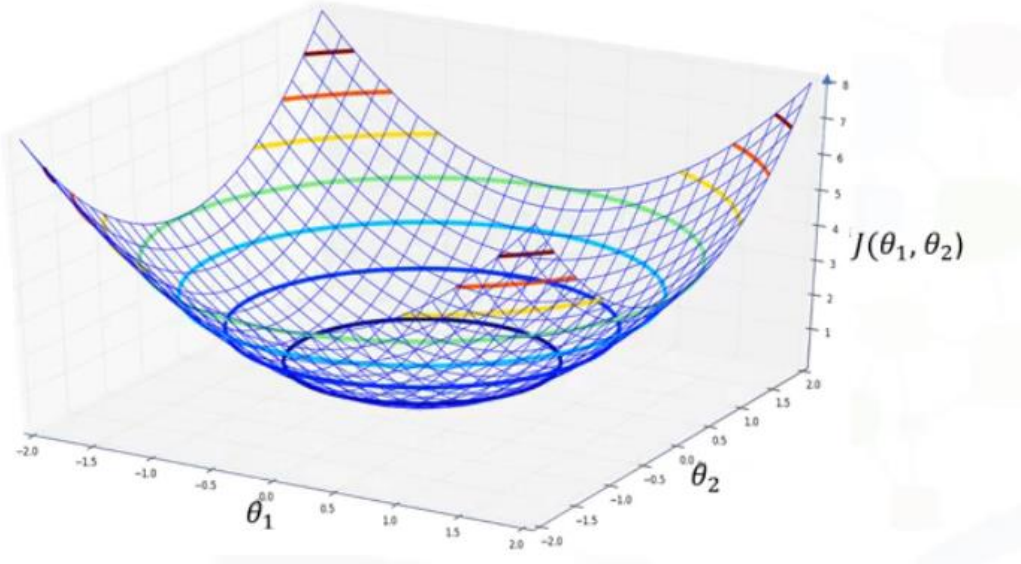
الشكل 2-8 دوال التفعيل

بَيْدَ أن أكثر الدوالّ استخدامًا في هذه الأيام هي (Rectified Linear Unit: ReLU)، وذلك للميّزة الأكثر أهمية وهي عدم تفعيل كل العصبونات في نفس الوقت، ما يُساهم في تقليل كمية الحسابات المُنجزة، إذ تُصفِّر القيم السالبة.

فهي من البساطة بمكان: كل القيم المساوية إلى الصفر أو أصغر منه تُصبح صفرًا، بينما تبقى القيم الموجبة كما هي.

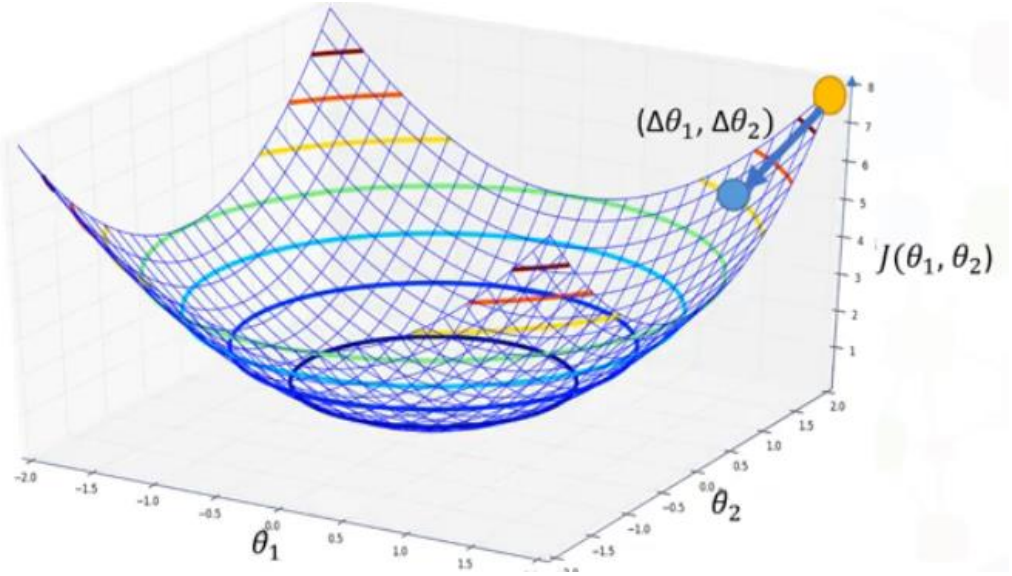
2-4-الانحدار التدريجي (Gradient Descent):

يعد الانحدار التدريجي من أبرز خوارزميات التعلم في الشبكات العصبونية، وهي منهجية تكرارية لإيجاد القيمة الاقل لتابع الكلفة، وفي حالتنا هي خوارزمية تقوم بتغيير قيم البارامترات لتقليل قيمة تابع الكلفة او الخطأ، ننتقل الآن لآلية اختيار القيمة الدنيا لتابع الخطأ باستخدام الانحدار التدريجي:



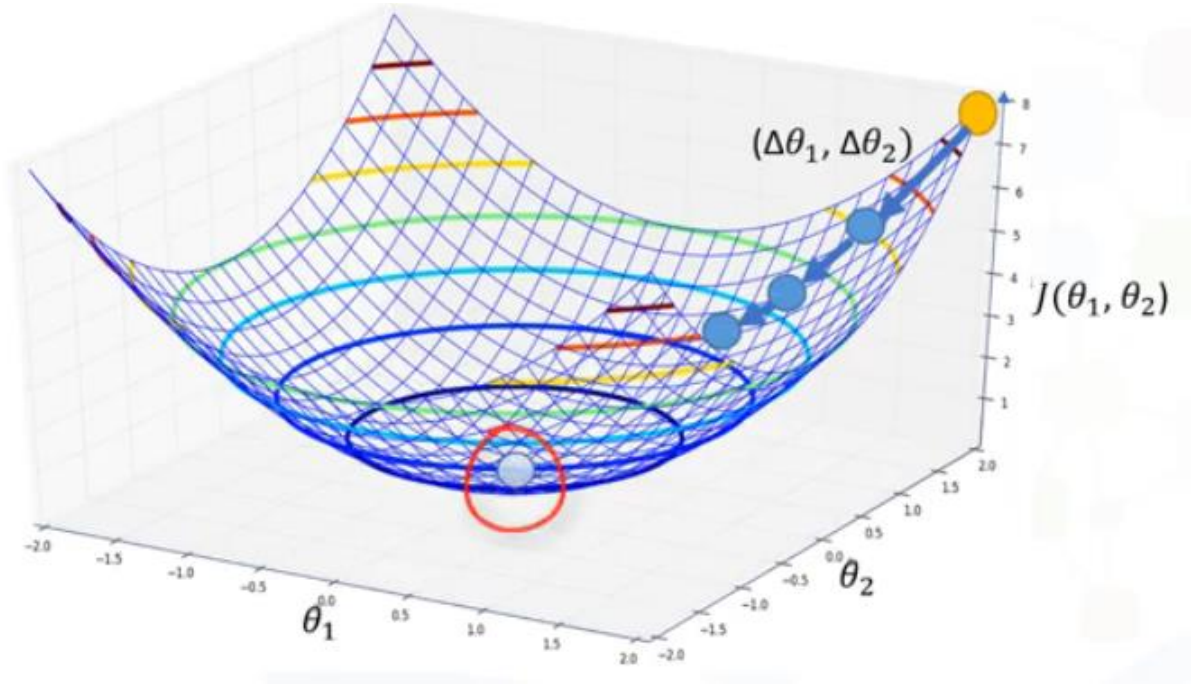
الشكل 2-9 منحني انحدار تدريجي تابع لبارامترين اثنين

ليكن لدينا المنحني الموضح بالشكل (2-9) والذي يمثل بمنحني الخطأ او حوض الخطأ المثل بسمتين والبعد الثالث لمراقبة تابع الكلفة لاختيار القيمة التي تحقق اقل تابع كلفة، ويمثل المنحني الموضح بالشكل (2-10) بدء تنفيذ الخوارزمية.



الشكل 2-10 قيمة تابع الكلفة الأعظمية

بداية نختار قيمة بارامتر عشوائية ثم نقوم بتغيير البارامترات ونأخذ خطوة للاتجاه نحو سطح المنحني لنحصل على اقل تابع كلفة، نستمر بأخذ الخطوات للوصول إلى سطح المنحني الممثل بقاع الحوض كما يمثل الشكل (2−11).



الشكل 2-11 قيمة تابع الكلفة الأصغرية

لمعرفة القيمة التي نخطو على اساسها نقوم بحساب التدرج، وهو هنا ممثل بميل السطح عن كل نقطة (مشتق المسافة بالنسبة للبارامترات) وتحدد قيمته مقدار الخطوة الواجب أن نقوم بها نحو القاع، في حال كان الميل كبير يجب ان تكون الخطوة كبيرة، وفي حال كان الميل صغير تكون الخطوة صغيرة، وتحدد إشارة المشتق اتجاه الحركة.

يوجد عدة خوارزميات تعلم مشابهة لمبدأ الانحدار التدريجي، هدفها تخفيض قيمة تابع الكلفة وبالتالي تخفيض قيمة الخطأ، وأهمها (The Adaptive Moment Estimation) أو (Adam optimization algorithm).

- Adam optimization algorithm:

هنالك العديد من الخوارزميات المُمكن تطبيقها للتدريب، إلا أن هذه الخوارزمية أثبتت كفاءَتها بالنسبة لطيفٍ واسعٍ من بُنى الشبكات العصبونية، كما يوصى بها كأداةٍ فعالة للأمْثَلَة العشوائية لكونها لا تحتاج غير بعض الانحدارات من الدرجة الأولى، وتقلّص من السعة التخزينية المطلوبة.

هي عبارة عن دمج بين الخوارزميتين: (Gradient Descent with Momentum: GDM) و(Root Mean Square Prop: RMSprop)، أي أن تابع الكلفة هنا هو الجذر التربيعي لمجموع مربعات الأخطاء.

فكرة التدريب: نبدأ بأوزان عشوائية، ونعدّل عليها بالاعتماد على كُل من الخرج المرغوب والخرج الذي تعطيه من خلال حساب تابع الخطأ والبحث عن النهاية الصُغرى لهذا التابع؛ أي أصغر قيمة ممكنة له في فضاء الأوزان.

إذن المسألة هي: إيجاد النهاية الصُغرى لتابع الخطأ؛ إذ تسعى الخوارزمية إلى إيجاد الوزان المُثلى، وتقليص الخطأ، ورفع الدقّة.

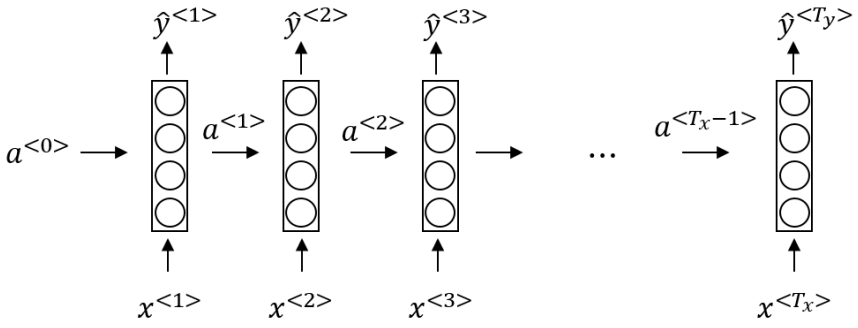
نبدأ بالانطلاق من الأوزان العشوائية وإيجاد النهاية الصُغرى لتابع الخطأ؛ أي البحث عن مجموعة الأوزان التي تجعله أصغر ما يُمكن.

لكن خوارزمية الانحدار التدريجي قد تصل بنا إلى نهاية صُغرى محلية (Local Minimum) لنعلق فيه بالرغم من وجود نهاية صُغرى أكثر انحدارًا (Global Minimum) هي المنشودة.

وللحؤول دون الوقوع في نهاية محلية أضفنا الـ Momentum (يُماثل مدلوله الفيزيائي؛ زخم الحركة في تقليل التفاوتات الكبيرة للأوزان من مثال لآخر)، وبالتالي نخطو خطوة إضافية أو دفعة للتأكد من عدم وجود نهاية صغرى ذات قيمة أفضل من القيمة الحالية.

2-5-الشبكات العصبونية التكرارية (Recurrent Neural Networks):

تحتفظ الشبكة العصبية المتكررة بالمعلومات الماضية وتتأثر قراراها بما تعلمته من الماضي. ففي حين تحتفظ شبكات التغذية الأمامية الأساسية بالمعلومات أيضًا، إلا أنها لا تتذكر الأشياء التي تعلمتها أثناء مرحلة التدريب، على سبيل المثال، يتعلم مصنف الصور الشكل " 1" أثناء التدريب ثم يستخدم تلك المعرفة لتصنيف الأشياء في مرحلة المنتج، بينما تتعلم RNNs بشكل مشابه أثناء التدريب، إلا أنها تتذكر الأشياء المستفادة أيضاً من المدخلات السابقة أثناء توليد المخرجات التالية، فهي جزء من عمل الشبكة. يمكن لشبكات RNN أن تأخذ شعاعاً واحداً أو أكثر من الأشعة وأن تنتج شعاعاً واحداً أو أكثر من الأشعة ولا تتأثر المخرجات الخاصة بها فقط بالأوزان المطبقة على المدخلات مثل NN العادية، ولكن أيضًا تتأثر بواسطة متجه الحالة "المخفية" الذي يمثل السياق استنادًا إلى المدخلات / المخرجات السابقة. لذلك، يمكن أن ينتج عن نفس المدخلات مخرجات مختلفة اعتمادًا على المدخلات السابقة في السلسلة. انظر الشكل (12-2):



الشكل 2-12 بنية الشبكة العصبونية التكرارية

3-الفصل الثالث: معالجة الإشارة الصوتية

3-1-الإشارة الصوتية وتكوين الصوت:

لفهم عملية التعرف على الجنس يجب علينا أولا أن نفهم بنية الصوت والكلام، إن الكلام الذي يتحدث به البشر هو الشكل الطبيعي للتواصل فيما بينهم والذي يتطلب استعمال الصوت. والصوت هو تردد آلي أو موجة قادرة على التحرك في عدة أوساط مادية مثل الأجسام الصلبة، السوائل والغازات، ولا تنتشر في الفراغ. وباستطاعة الكائن الحي تحسسه عن طريق عضو خاص يسمى الأذن.

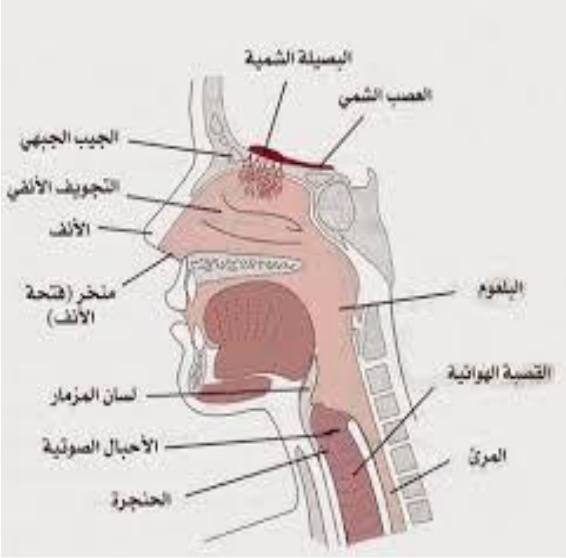
من منظور علم اللغة فالصوت هو إشارة تحتوي على نغمة أو عدة نغمات تصدر من الكائن الحي الذي يملك العضو الباعث للصوت، وتستعمل كوسيلة اتصال بينه وبين كائن آخر من جنسه أو من جنس آخر، يعبر من خلالها عما يريد قوله أو فعله بوعي أو بغير وعي مسبق، ويسمى الإحساس الذي تسببه تلك الذبذبات بحاسة السمع.

ويعد الصوت أساس الكثير من الخبرات التي يكتسبها الانسان، وقد كان الانسان في الماضي ل يعتمد على الأصوات التي يصدرها من حنجرته فحسب، وإنما أيضا على أصوات الطبول والأدوات التي تحدث الجلجلة والخشخشة وأيضا المزامير.

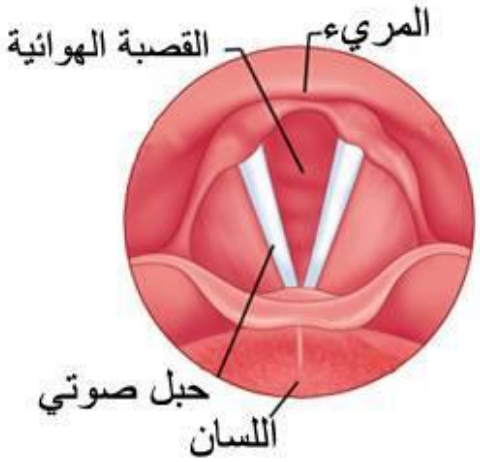
وتقدر سرعة الصوت في وسط الهواء العادي ب 343 متر في الثانية أو 1224 كيلو متر في الساعة. وتتعلق سرعة الصوت بعامل لصلابة وكثافة المادة التي يتحرك فيها الصوت.

هناك ثلاثة أجزاء رئيسية في جسم الانسان تقوم بلعب الأدوار المطلوبة لإنتاج الصوت وهي الرئتين، الحنجرة والحبال الصوتية، الأنف والفم.

تبدأ هذه العملية من الرئتين ومنهما يخرج الهواء منطلقا إلى الحنجرة حيث يمر بالحبال الصوتية التي هي المرحلة الثانية في عملية إنتاج الصوت وهي أهم الأجزاء الثلاثة، والأحبال الصوتية هي عبارة عن زوجين من الأغشية المخاطية الممتدة بشكل عرضي في الجزء الأوسط من الحنجرة حيث يوضح الشكل (3-1) موقع الحبال الصوتية من الحنجرة، وفي الشكل (3-2) يمكن رؤية كيف تظهر الأحبال الصوتية للناظر إلى داخل الحنجرة من الأعلى.



الشكل 3-1 موقع الحبال الصوتية مع الحنجرة



الشكل 3-2 الحبال الصوتية كما تظهر من الأعلى

تكون الحبال الصوتية في وضعها الطبيعي مفتوحة وتقترب من بعضها في حال التحدث على حسب الصوت المراد إصداره كما يبين الشكل (3-3)

الشكل 3-3 الحبال الصوتية في وضع السكون والتحدث

3-2-مجموع الالتفاف (Convolution Summation):

هو عملية رياضية بين تابعين لتشكيل تابع ثالث يعتمد قيمتيهما حيث يحسب مجموع الالتفاف بحساب مجموع جداءات كل قيمة من أحد التابعين مع قيمة مزاحة من التابع الآخر كما تبين المعادلة التالية:

المعادلة (3-1):

$$(f \otimes g = \sum_{m=-N}^{+N} f(m) * g(n-m)$$

حيث أن الرمز $\otimes$ يدل على مجموع الالتفاف.

إن مجموع الالتفاف ليس له أي معنى فيزيائي مباشر إنما يستخدم في حساب ناتج ربط تابعين رياضيين مع بعضهما البعض، حيث أن هذا المجموع يستخدم في مجالات متعددة (الاحتمالات، معالجة الإشارة...)

في معالجة الإشارة نستخدم مجموع الالتفاف لحساب الإشارة الناتجة عن استجابة وسط ما للطاقة المقدمة لهذا الوسط، كما يستخدم في عملية تقطيع الإشارة إلى إطارات كما سنرى لاحقا.

3-3-تحويل فوريه (Fourier Transform):

يقدم تحويل فوريه أسلوبا لحساب الترددات المكونة لإشارة معينة حيث تم إثبات أن أي إشارة أو تابع رياضي يمكن أن يكتب كمجموع توابع جيبية وكل من هذه التوابع له تكرار معين يسمى التردد.

إن تحويل فوريه يقيس التشابه بين الإشارة أو التابع قيد الدراسة وبين مجموعة التوابع الجيبية ذات الترددات المختلفة كما توضح المعادلة التالية:

المعادلة (3-2):

$$F(\omega) = \sum_{n=-\infty}^{+\infty} f(n)e^{-i\omega n}$$

حيث F(ω) هو تحويل فوريه للإشارة

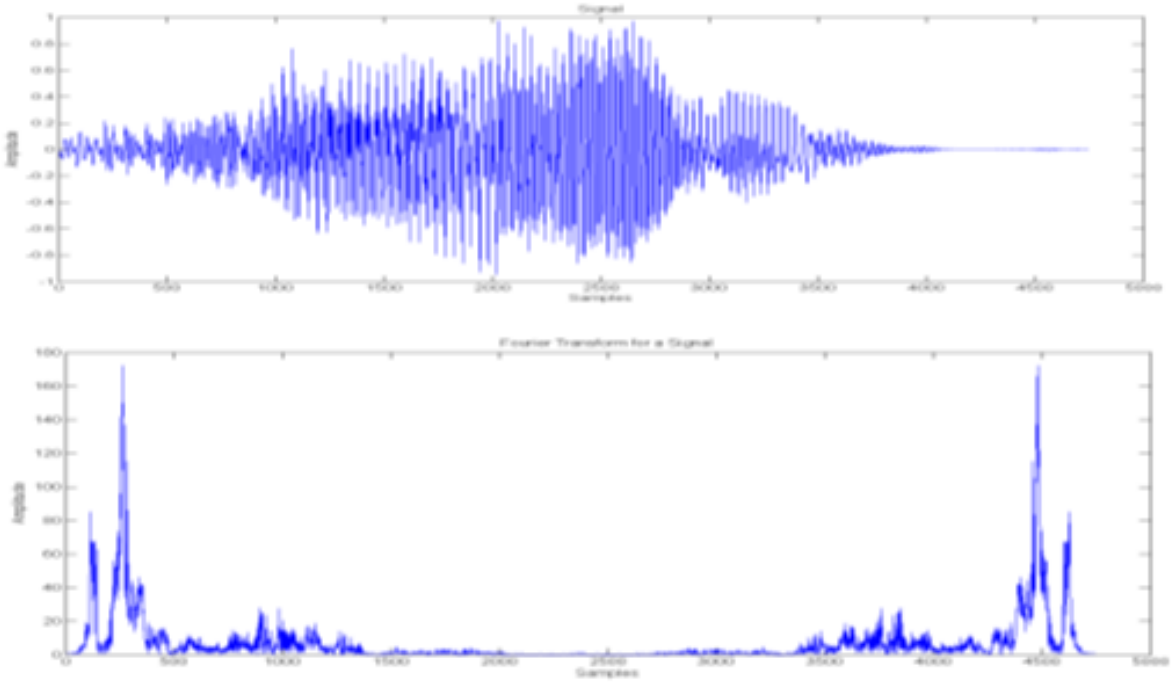
f(n) عينات الإشارة في اللحظة n

و ω تمثل تردد الإشارة التي نرغب بحساب توافقها مع هذه الإشارة.

نلاحظ من المعادلة السابقة أن ناتج تحويل فوريه هو عبارة عن أعداد عقدية تمثل نسبة التوافق بين هذه الإشارة والتابع الجيبي الموافق للتردد n حيث يمثل مطال هذه الأعداد العقدية مقدار التوافق بين الإشارتين، وتسمى هذه المطالات طيف الإشارة الترددي.

نلاحظ أننا نقلنا الإشارة من مجال الزمن إلى مجال التردد، حيث أن مجال الزمن هو تغير الإشارة بتقدم الزمن وكل لحظة زمنية تمثل مطال معين، أما مجال التردد فهو تغير استجابة الإشارة بزيادة التردد حيث في الإشارات ذات التردد المنخفض تكون القيم المرتفعة للترددات منخفضة المطالات في طيف الإشارة والعكس بالعكس.

إن أهم فوائد تحويل فوريه أنه يساعد في معرفة ترددات الإشارة حيث أنه تكون قيم مطالات الطيف الترددي كبيرة عند الترددات الأكثر فعالية في هذه الإشارة وضعيفة عند باقي الترددات وبالتالي يمكن تحديد تردد الإشارة والترددات غير المرغوبة والتي تعتبر ضجيج على الإشارة.



الشكل 3-4 إشارة صوتية تابعة للزمن والطيف الترددي الخاص بها

إن تحويل فوريه متناظر أي أن القيم بعد منتصف الطيف تماثل القيم قبل منتصف الطيف كما يبين الشكل (3-4) لذلك يمكن الاحتفاظ فقط بالنصف الأول من التحويل.

نلاحظ من الشكل (3-4) أن أغلب الاستجابات تحدث في مجال الترددات المنخفضة أي أن تردد الإشارة وأكبر استجابة هي بتردد 250 Hz تقريبا لذلك نجد أن لتحويل فوريه فائدة في تحديد الترددات ذات الأثر الأكبر في الإشارة.

في تحويل فوريه إن طيف الإشارة هو عبارة عن جداء طيف الإشارة الصادرة عن المنبع مع طيف استجابة الوسط، أي نستطيع القول أنه بتحويل الإشارة إلى مجال التردد نحول مجموع الالتفاف إلى جداء أطياف الإشارات المشكلة لهذه الإشارة كما تبين المعادلتين التاليتين:

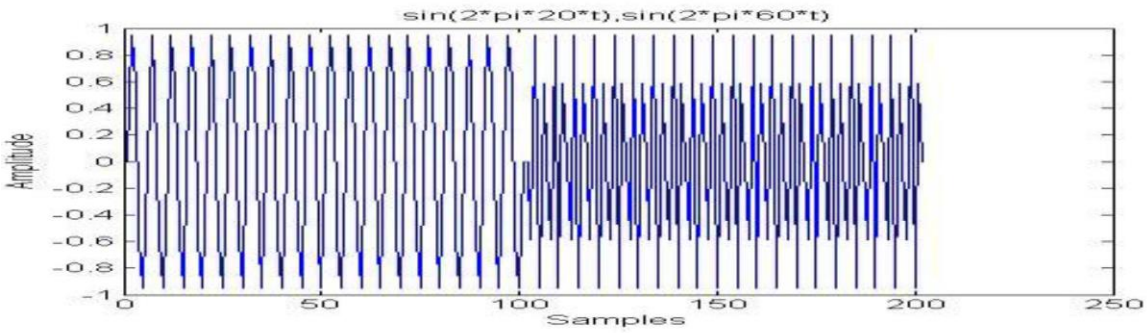
المعادلة (3-3):

$$s(t) = \sum_{n=-\infty}^{+\infty} e(n)h(t-n)$$

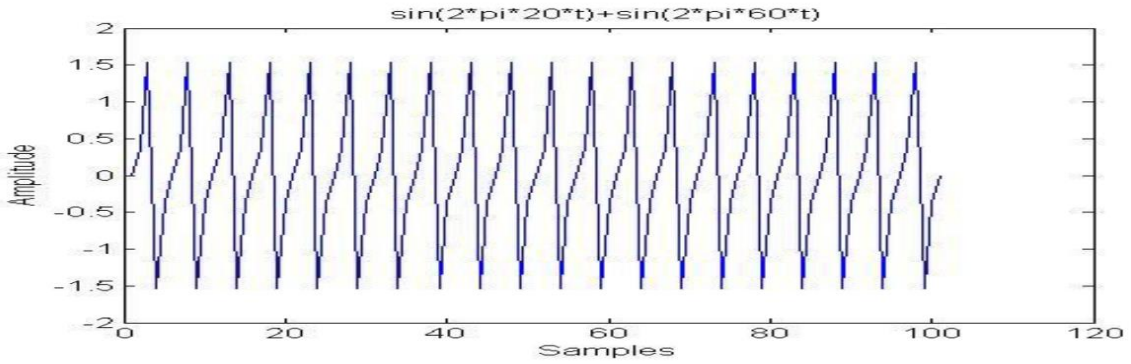
حيث H هو طيف الإشارة المقابلة لاستجابة الوسط، E طيف الإشارة الصادرة من المنبع.

على الرغم من قدرة تحول فوريه على تحديد الترددات المكونة للإشارة إلا أن هناك عيبين رئيسيين في هذا التحويل:

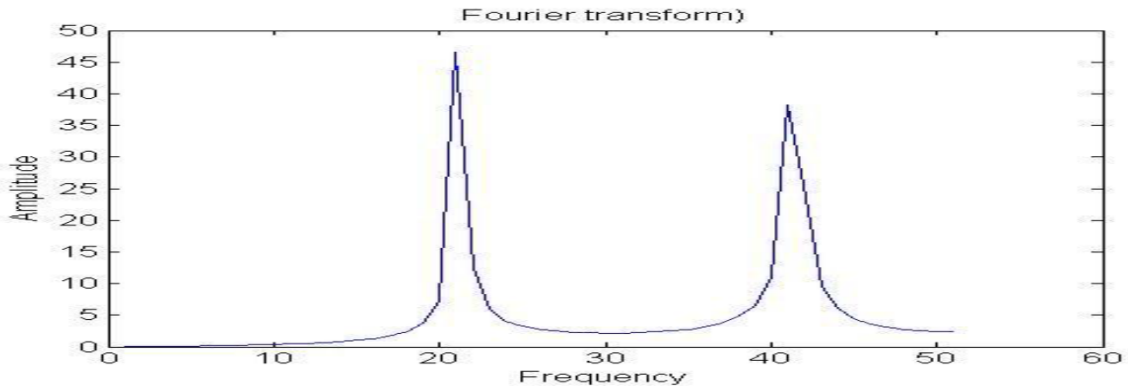
- العيب الأول: إن تحويل فورييه يحدد ترددات الإشارة ولكن لا يحدد في أي اللحظات الزمنية حصلنا على هذا التردد حيث أن تحديد اللحظات الزمنية له أهمية كبيرة في معرفة تغير تردد الإشارة مع الزمن.
- العيب الثاني: لنفرض لدينا الإشارات الجيبية التالية بترددات 20, 60 Hz حيث يمثل الشكل (3-5) إشارتين جيبيتين بترددين مختلفين ويمثل الشكل (3-6) مجموع الإشارتين الجيبيتين بنفس الترددين ويمثل الشكل (3-7) تحويل فورييه لكل من الإشارتين الموضحتين في الشكلين (3-5) و (3-6)



الشكل 3-5 إشارتان جيبيتان بترددين مختلفين



الشكل 3-6 مجموع الإشارتين الجيبيتين بنفس الترددين



الشكل 3-7 تحويل فورييه لكل من الإشارتين السابقتين

3-4-التقطيع إلى إطارات ونوافذ (Framing and Windowing):

تقطيع الإشارة هو عملية تقسيم الإشارة إلى عدد من الأقسام يسمى كل منها إطار ويعالج كل إطار بشكل مستقل عن بقية الإطارات.

في أغلب الأحيان لا نستطيع التعامل مع الإشارة دفعة واحدة لأن ذلك قد يسبب نتائج غير مرضية حيث كما وجدنا الفقرة السابقة أن تحويل فوريه لا يحدد اللحظات الزمنية الموافقة لتردد معين لذلك كان التقطيع هو الحل الأفضل لحساب الترددات لشكل شبه محلي أي نكوّن معرفة جزئية عن اللحظات الزمنية الموافقة للترددات.

أما تطبيق النوافذ أو النوفذة Windowing في عملية حساب تشابه بين تابعين رياضيين حيث يكون الأول هو الإشارة أو الإطار المدخل والثاني هو التابع الرياضي المعبر عن النافذة أي أن تطبيق النافذة هو مجموع التفاف بين الإشارة والنافذة ويعبر عنها بالعلاقة:

المعادلة (3-4):

$$s(n) = \sum_{i=n-m+1}^{n} x(i)w(n-i)$$

توجد أنواع عديدة جدا من النوافذ وكل هذه النوافذ (عدا المستطيلة) نميل لأن تكون منخفضة المطالات عند طرفي النافذة ومرتفعة في المنتصف وذلك لإلغاء تأثير عينات الإشارة من الطرفين وحصر تطبيق النافذة بمجموعة من العينات في وسط النافذة. سنتكلم عن ثلاث أنواع من النوافذ وهي المستطيلة، هامينغ والمثلثية، سنستعرض أهم أنواع النوافذ.

3-4-1-النافذة المستطيلة (Rectangular Window):

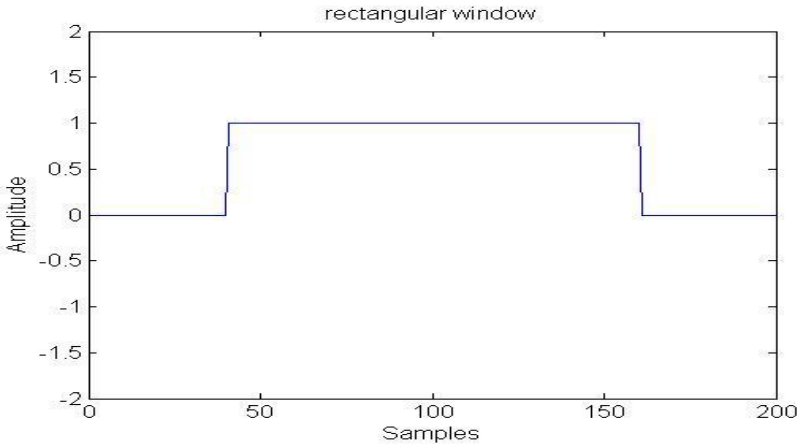
وتعتبر أبسط النوافذ ويعبر عنها بالمعادلة التالية:

المعادلة (3-5):

$$\mathrm{w(n)}=\begin{cases}1, & |n| \le N/2\\ 0, & n > N/2\end{cases}$$

حيث N هو طول النافذة.

إن تطبيق هذه النافذة على إشارة معينة يكافئ تقطيع هذه الإشارة إلى إطارات كل منها بطول N وهذه النافذة تمثل بالشكل كما يظهر بالشكل (3-8):



الشكل 3-8 النافذة المستطيلة

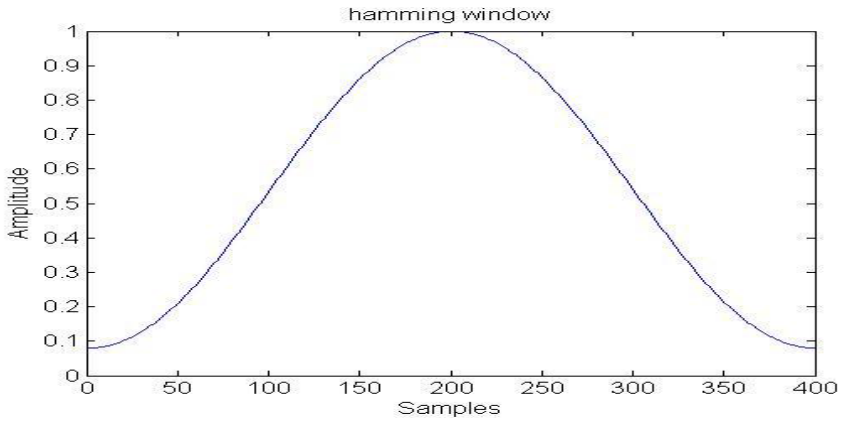
3-4-2-نافذة هامينغ (Hamming Window):

هي أشهر أنواع النوافذ ولها استخدامات كثيرة في مجال معالجة الإشارة كما سنرى في هذا البحث حيث أن هذه النافذة تعطى بالمعادلة:

المعادلة (3-6):

$$w(n) = 0.54 - 0.46 * \cos(2\pi * \frac{n}{N-1})$$

حيث N طول النافذة، وشكل هذه النافذة كما يظهر بالشكل (3-9):



الشكل 3-9 نافذة هامينغ

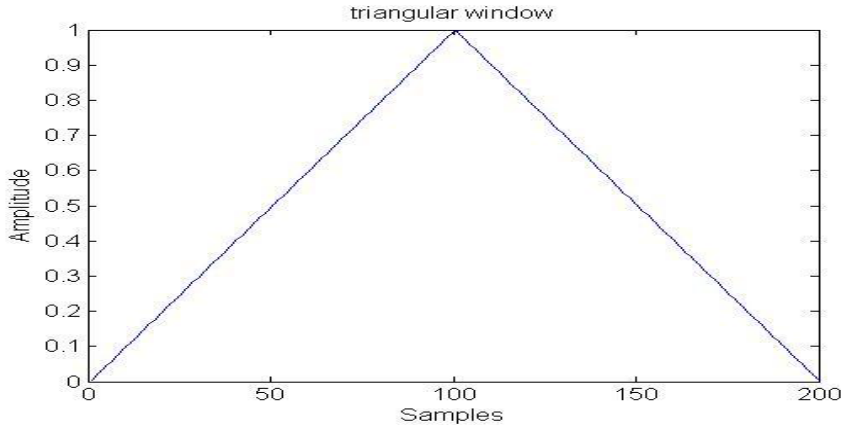
3-4-3-النافذة المثلثية (Triangular Window):

تعتبر هذه النافذة هي مجموع التفاف بين نافذتين مستطيلتين ويعبر عنها بالمعادلة:

المعادلة (3-7):

$$w(n) = 1 - \left| \frac{n - \frac{N-1}{2}}{\frac{N}{2}} \right|$$

حيث N طول النافذة، وشكل هذه النافذة كما يظهر بالشكل (3-10):



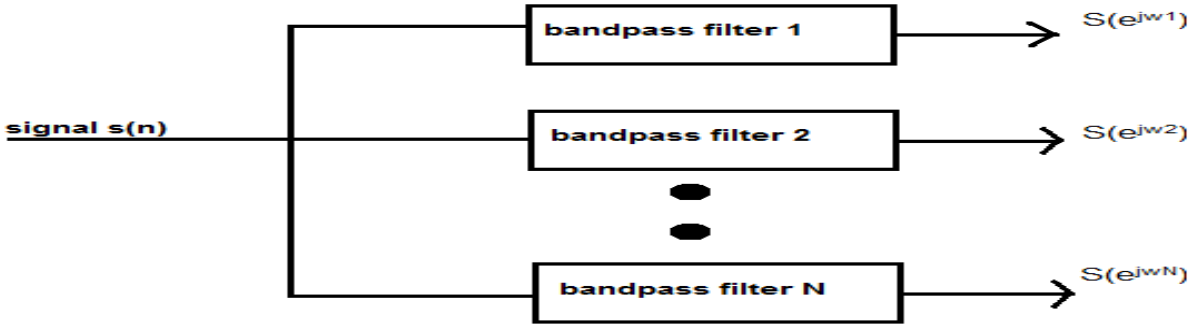
الشكل 3-10 النافذة المثلثية

3-5-المرشحات (Filter Bank):

هي عملية تطبيق مجموعة من المرشحات الترددية على الإشارة بهدف حساب استجابة الإشارة لكل من المرشحات، وتعتبر هذه العملية من أهم العمليات في معالجة الإشارات حيث أنها تساعد في تحديد مركبات الإشارة وتحليل بناء الإشارات.

تعتبر هذه النظم ذات تقييمات متعددة multi-rate systems وذلك لتنوع النتائج التي تظهر من تطبيق مجموعة من الترددات على الإشارة والتي تؤدي إلى نتائج مختلفة لنفس الإشارة.

يعبر الشكل (3-11) عن تطبيق المرشحات:



الشكل 3-11 تطبيق المرشحات على الإشارة

إن المرشحات الترددية ما هي إلا عملية تطبيق لنوافذ على الإشارة لحساب التشابه بين هذه النافذة والإشارة ولكن الفرق بين النوافذ وتطبيق المرشحات هو أنه بتطبيق النوافذ تمرر على عينات الإشارة نافذة واحدة أما في المرشحات يطبق على الإشارة مجموعة من النوافذ لكل منها قيم مختلفة عن قيم الأخرى وهذا ما يجعل النوفذة (تطبيق النوافذ) لا يحدد الاستجابات الترددية للإشارة حيث يعبر عن تطبيق المرشحات بمجموع الالتفاف التالي:

المعادلة (3-8):

$$s_i(n) = s(n) \otimes h_i(n) = \sum_{m=0}^{M-1} s(m) * h_i(n-m) \; where \; 1 \leq i \leq N$$

حيث i ترتيب المرشح، hi استجابة المرشح، M طول نافذة المرشح، si الإشارة المرشحة.

إن المرشحات في بحثنا هي مرشحات تمرير حزمو أي أن كل مرشح خاص بمجموعة معينة من الترددات ويتم حساب طيف الإشارة في هذا المرشح تبعا لترددات.

- أنواع المرشحات حسب طول النافذة:

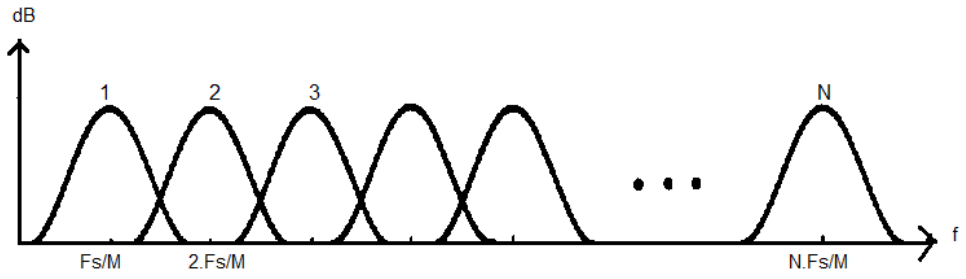
إن لكل مرشح تردد مركزي متساو البعد عن الترددين الأعظمي والأصغري لهذا المرشح، حيث أن المسافة بين الترددات المركزية للمرشحات تحدد أطوال نوافذ المرشحات وبالتالي نوع المرشحات حيث تقسم إلى قسمين رئيسيين:

1. المرشحات المنتظمة Uniform Filter Bank: يعتبر هذا النوع هو النوع الأقدم من المرشحات حيث تكون النوافذ متساوية الطول كما يظهر في الشكل (3-12) ويكون التردد المركزي لكل مرشح يحسب من العلاقة:

المعادلة (3-9):

$$f_i = \frac{f_s}{M}.i \quad where\ 1 \leq i \leq N$$

حيث N هو عجج المرشحات، M عدد المرشحات اللازمة لتغطية كامل المجال الترددي Fs تردد تقطيع الإشارة وبشكل عام إن N ≤M/2، وعرض مجال المرشح هو Fs / M



الشكل 3-12 المرشحات المنتظمة

1. المرشحات غير المنتظمة Non Uniform Filter Bank: في هذا النوع من المرشحات يكون عرض كل مرشح مختلف عن باقي المرشحات وهذا الأمر يسمح بالتركيز على استجابة مرشح دون الآخر، ويعتبر هذا النوع مفيد جدا في أنظمة التعرف على الجنس حيث يمكن بناء مرشحات تحاكي استجابة الأذن لترددات الصوت.

يمكن التعبير عن عرض هذه المرشحات بأي علاقة رياضية متغيرة القيم حيث في المعادلة التالية يزداد عرض حزمة المرشح بشكل خطي:

المعادلة (3-10):

$$b\_1=C$$
$$b\_i= \alpha.b\_(i-1)$$

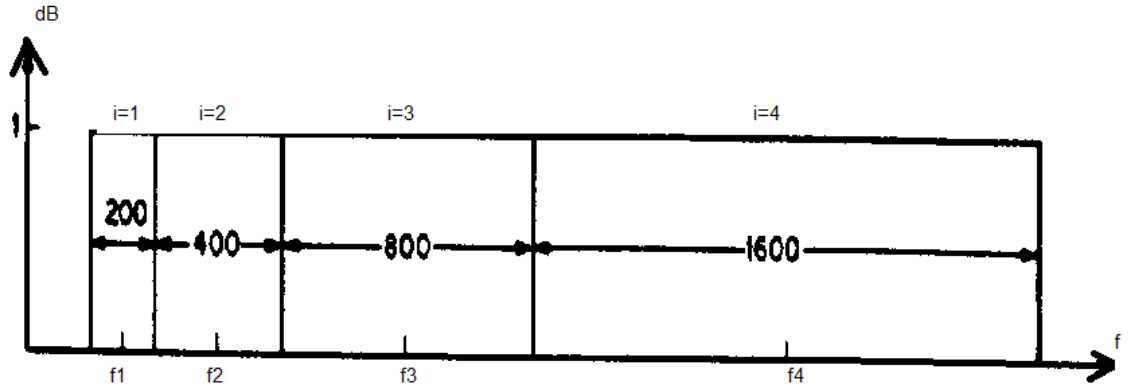
حيث C ثابت وهو عرض حزمة المرشح الأول، α مقدار الزيادة.

والتردد المركزي لهذه المرشحات يعطى كما يلي:

المعادلة (3-11):

$$f_i = f_1 + \sum_{j=1}^{i-1} b_j + \frac{b_i - b_1}{2}$$

بفرض C=200، α=i نحصل على الشكل (3-13):



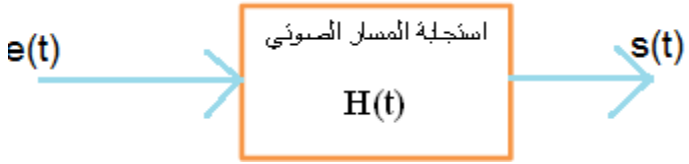
الشكل 3-13 المرشحات غير المنتظمة المتزايدة بشكل خطي

3-6-تحليل المركبات الطيفية (Cepstral Analysis):

إن كلمة cepstral ليس لها معنى لغوي إنما هي تركيب معكوس للأحرف الأولى من كلمة Spectral (طيفي)، وذلك للدلالة على أن هذه العملية تطبق على طيف الإشارات الترددي لاستخراج مجموعة المركبات التي تمثل هذه الإشارة، وكذلك الحال بالنسبة لكلمة cepstrum هي معكوس لكلمة spectrum.

إن أي إشارة هي عبارة عن استجابة وسط معين لكلمات من الطاقة التي يخضع لها، فالإشارة الصوتية هي استجابة المسار الصوتي للطاقة الناتجة عن ضغط الهواء الناتج عن الرئتين.

إن استجابة المسار h(t) الصوتي لهذه الطاقة e(t) هو ما يحدد شكل الإشارة الناتجة وبالتالي الصوت s(t) الناتج phoneme كما في الشكل (3-14):



الشكل 3-14 تشكيل الإشارة s(t) من تركيب الإشارة الممثلة للطاقة e(t) مع استجابة المسار الصوتي h(t).

يتم تمثيل هذه الإشارة رياضيا باستخدام الالتفاف convolution حيث أن الإشارة الناتجة هي مجموع التفاف الإشارتين كما في المعادلة:

المعادلة (3-12):

$$s(t) = e(t) \times h(t)$$

إن الهدف من تحليل أي إشارة هو الحصول على استجابة الوسط h(t) حيث أن هذه الاستجابة هي التي تحدد ما هو الصوت المنطوق.

لذلك نحتاج إلى فصل استجابة المسار الصوتي عن طاقة الإشارة، تعتبر هذه العملية صعبة التطبيق في مجال الزمن بسبب مجموع الالتفاف convolution لذلك نحتاج إلى تحويل هذا المجموع إلى شكل يمكن استخلاص الاستجابة منه فنستخدم تحويل فوريه لنقل الإشارة إلى مجال التردد ونطلق اسم الطيف الترددي للإشارة على خرج تحويل فوريه الذي يعطى بالمعادلة:

المعادلة (3-13):

$$S(f) = E(f).H(f)$$

إن استجابة الوسط (المسار الصوتي) هي استجابة بطيئة التغير في الزمن وبالتالي هي إشارة ذات تردد منخفض، وطاقة الهواء المندفع في هذا الوسط هي إشارة سريعة التغير وبالتالي ذات تردد مرتفع فبالإمكان تحويل الضرب إلى جمع خطي لكل من الإشارتين وذلك بتطبيق لوغاريتم الطرفين كما في المعادلة:

المعادلة (3-14):

$$|\log(S(f))| = |\log(E(f).H(f))| = |\log(E(f))| + |\log(H(f))|$$

إن هذا الجمع الخطي للوغاريتم الإشارتين يجعل بالإمكان الحصول على استجابة الوسط والذي نحصل عليه بتطبيق تحويل فوريه العكسي على المعادلة (3-3) فنحصل على المركبات الطيفية الممثلة لإشارة الصوت المدخل.

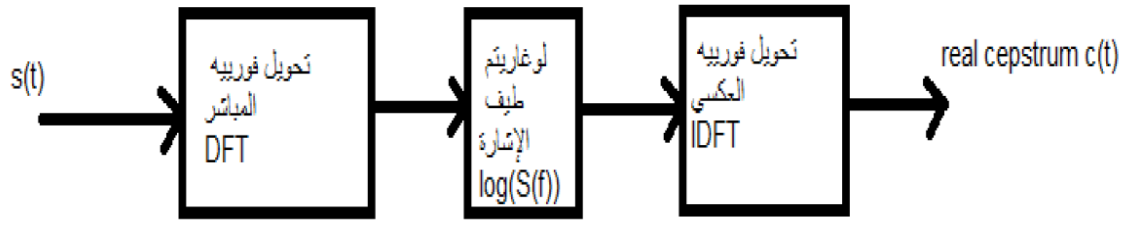
أي أنه بتطبيق تحويل فوريه العكسي على لوغاريتم الطيف الترددي للإشارة S(f) نحصل على المركبات الطيفية الممثلة لهذه الإشارة والتي نطلق عليها real cepstrum هذه الإشارة أي أن القسم الحقيقي من هذا تحويل فوريه العكسي هو هذه المركبات.

يمكن تلخيص كل العمليات السابقة بالمعادلة:

المعادلة (3-15):

$$C_s(n) = \frac{1}{N}\sum_{k=0}^{N-1} log|S(k|.e^{-j2\pi kn/N} \; for \; n = 0,1,\dots,N-1$$

أي أن التحليل الطيفي للمركبات cepstral analysis هو محاولة الحصول المركبات التي تمثل استجابة الوسط للطاقة المقدمة من المنبع حيث تعتبر هذه المركبات هي القسم الحقيقي من تحويل فوريه العكسي المطبق على لوغاريتم الطيف الترددي للإشارة وذلك كما يوضح الشكل التالي(3-15).



الشكل 3-15 مراحل تحليل المركبات لتمثيل الإشارة للحصول على المركبات الممثلة لاستجابة الإشارة

3-7-خوارزمية Mel Frequency Cepstral Coefficients (MFCC) لاستخراج السمات:

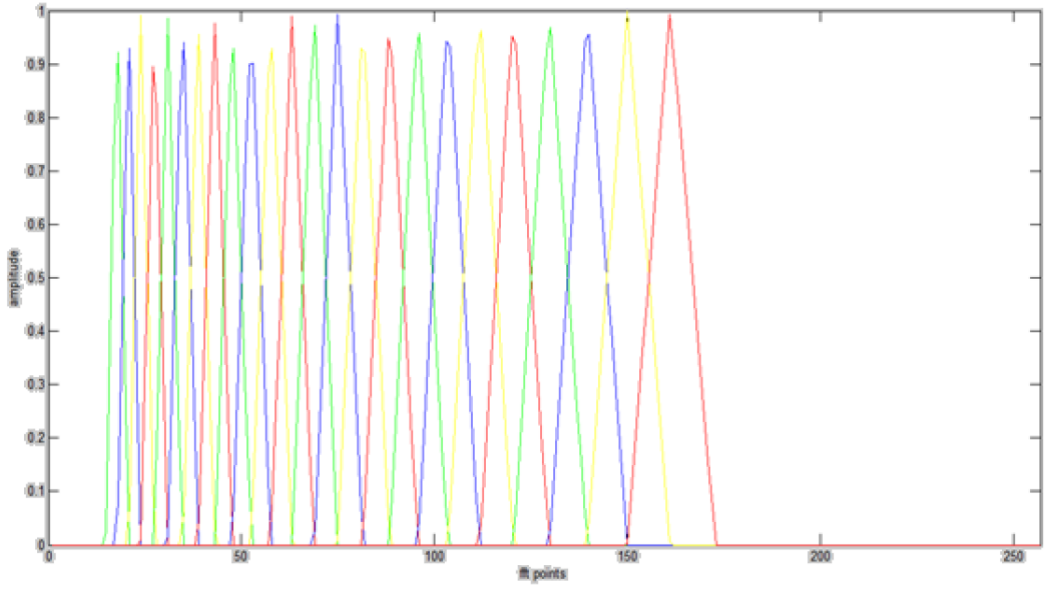
تعتبر هذه الخوارزمية من أكثر الخوارزميات شيوعا في استخراج السمات المميزة للصوت في نظم التعرف على الكلام وذلك بسبب دقة نتائجها والقدرة على التخلص الجزئي من ضجيج الإشارة وكذلك سرعة تطبيقها وسهولته. كما أنه في هذه الخوارزمية تتم مقاربة عملية السمع عند الإنسان أي محاولة استخراج سمات الإشارة بشكل يتوافق تقريبا مع آلية السمع عند الإنسان حيث أن الأذن البشرية حساسة للترددات المنخفضة تحت 1000 Hz وضعيفة الحساسية للترددات المرتفعة فوق 1000 Hz، لذلك فإن هذا السلوك جعل بالإمكان تطبيق مجموعة معينة من المرشحات على الإشارة لكل مرشح مجال ترددي معين.

تسلك هذه المرشحات سلوكا خطيا من أجل الترددات المنخفضة وسلوكا لوغاريتميا من أجل الترددات المرتفعة، أي تتم زيادة عرض المجال الترددي للمرشح بشكل خطي من أجل الترددات 1000 Hz وبشكل لوغاريتمي للترددات فوق 1000 Hz، والسبب في تصميم المرشحات بهذا الشكل هو أن الأذن غير حساسة للترددات المرتفعة وبالتالي يمكن تقليل عدد المرشحات المميزة لهذه الترددات.

يطلق على هذه المرشحات Mel Scale وتكون عبارة عن مجموعة من الإشارات المثلثية كما يبين الشكل (3-16)، تمثل كل إشارة مرشحا لمجال ترددي مختلف، ويمكن التعبير عن مراكز هذه المرشحات بالمعادلة:

المعادلة (3-16):

$$F_{mel} = 2595 \times log_{10}\left(1 + \frac{f_{Hz}}{700}\right)$$

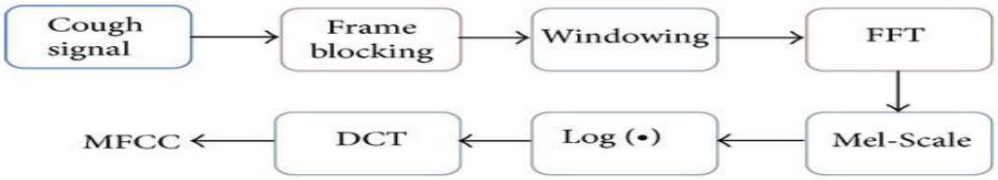


الشكل 3-16 المرشحات الترددية المستخدمة في MFCC والتي تسمى Mel Scale

تعتمد خوارزمية MFCC على أسلوب التحليل الطيفي للمركبات Cepstral Analysis التي تعرفنا عليها سابقا بالإضافة إلى تطبيق مجموعة من المرشحات التي تحدد استجابة الإشارة من أجل مجالات ترددية

مختلفة وبالتالي يمكن حصر مجال ترددات الإشارة ومركباتها في المرشحات التي تكون استجابة الإشارة لهذه المرشحات مرتفعة، وتصبح السمات اللازمة لتمثل هذه الإشارة أقل.

في MFCC تمر الإشارة الصوتية بعدد من المراحل التي تكون نتيجتها هي السمات المميزة لهذه الإشارة، إن هذه المراحل تتم بالترتيب الموضح في الشكل (3-17):



الشكل 3-17 مخطط يوضح مراحل عمل خوارزمية MFCC لاستخراج السمات

ننتقل الآن لدراسة خطوات خوارزمية MFCC.

3-7-1-تقسيم الإشارة إلى إطارات (Frame Blocking):

إن أول مرحلة في استخراج السمات باستخدام MFCC هي تقسيم الإشارة إلى إطارات، يكون الدخل هو الإشارة الصوتية والخرج هو الإطارات الموافقة لهذه الإشارة، حيث أن كل إطار من هذه الإطارات يعالج بشكل مستقل عن بقية الإطارات والهدف الأساسي من عملية التقسيم هذه هو حساب السمات الموافقة لتغير الإشارة بالنسبة للفاصل الزمني الذي يحدده طول الإطار، أي بعبارة أوضح إن كل إطار يقابل مجموع من عينات الإشارة فيتم حساب السمات الموافقة لهذه العينات وبالتالي تتم مراعاة تغيرات الإشارة مع الزمن.

يتم تقسيم هذه الإشارة تبعا لطول إطار معين مع حساب تداخل بين الإطارات وذلك لمراعاة ارتباط بعض العينات ببعضها البعض والتي قد تتبع لنفس المقطع الصوتي في كلمة معينة.

3-7-2-تطبيق النافذة (Windowing):

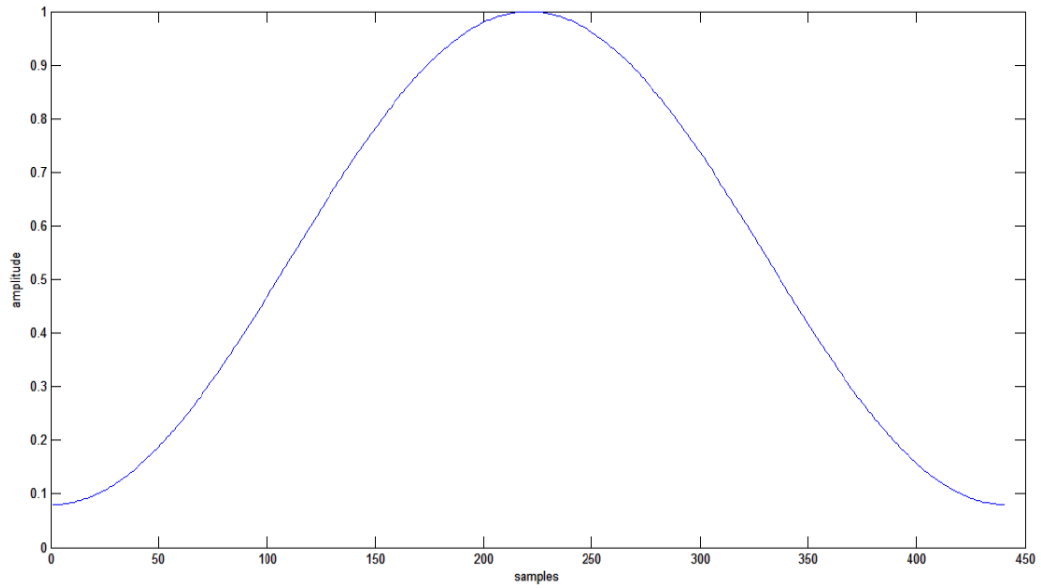
إن دخل هذه المرحلة هو الإطارات الناتجة من مرحلة التقسيم وخرجها هو الإطارات بعد تطبيق النافذة عليها، حيث يتم تطبيق نافذة على كل إطار وتكون هذه النافذة ذات قيم منخفضة عند الحواف ومرتفعة في المنتصف وذلك لحصر التغيرات الطيفية في مجال معين ومنع التشوهات التي قد تهر عند أطراف النافذة أي يتم التركيز على طيف الإشارة الموافق لمنتصف النافذة.

إن أفضل النوافذ وأكثرها استخداما في MFCC هي نافذة hamming المبينة بالشكل (3-18) والتي تعطى بالمعادلة:

المعادلة (3-17):

$$\begin{cases} 0.54 - 0.46 \times cos\left(\frac{2\pi n}{N-1}\right) & for\ n = 1,2,\dots,N-1 \\ 0 & otherwise \end{cases}$$

حيث N هي عدد عينات النافذة.



الشكل 3-18 نافذة هامينغ من أجل طول إطار 40 ms

3-7-3-تحويل فورييه السريع Fast Fourier Transform (FFT):

تعرفنا على تحويل فورييه وخصائصه في الفصل السابق (الأدوات المستخدمة في هذا البحث)، نطبق تحويل فورييه على خرج النوافذ لحساب الطيف الترددي لكل إطار والذي يعتبر دخل مرحلة المرشحات الترددية.

3-7-4-تطبيق المرشحات الترددية (Mel Frequency Warping):

يمكن اعتبار المراحل الثلاث السابقة مشتركة بين أغلب تقنيات استخراج السمات الصوتية، حيث تعتبر هذه المرحلة هي البداية الفعلية لاستخراج السمات باستخدام MFCC.

كما سبق وأوضحنا، إن خوارزمية MFCC تحاول محاكاة آلية استجابة الأذن للترددات حيث أن الأذن حساسة للترددات المنخفضة بشكل أكبر من حساسيتها للترددات المرتفعة، لذلك ومن أجل كل الطيف الترددي لكل إطار نقوم بتطبيق مجموعة من المرشحات الترددية (filter bank) على هذا الطيف بغية حساب استجابة هذا الطيف لهذه المرشحات، حيث أن الاستجابة هي مقدار التوافق بين المرشح والإشارة وهي حاصل ضرب قيم الترددات الممثلة للإشارة مع القيم الممثلة للمرشح وكلما زاد هذا الجداء كان التوافق أكبر، بكلمات أخرى الاستجابة هي طاقة الإشارة في هذا المرشح.

إن هذه المرشحات يمكن أن تمثل بأي تابع رياضي ولكن الأفضل ف MFCC أن يكون المرشح مثلثي الشكل كما هو موضح بالشكل (3-16)، ولكل مرشح تردد مركزي يعطى حسب المعادلة:

المعادلة (3-18):

$$F_{mel} = 2595 \times log_{10}\left(\frac{f_{Hz}}{700}\right)$$

حيث أنه ومن خلال هذا التردد يتم تحديد عرض المرشح أي مجال الترددات التي يسمح بتمريرها.

إن خرج هذه المرحلة هو الاستجابات الترددية للمرشحات التي تعتبر دخلا للمرحلة التالية وهي المرحلة التي يتم فيها فصل طاقة الإشارة عن استجابة المسار الصوتي.

3-7-5-تطبيق اللوغاريتم على خرج المرشحات Log Energy Computation:

إن MFCC خوارزمية تعتمد على التحليل الطيفي للمركبات Cepstral Analysis لاستخراج السمات كما أشرنا وشرحنا سابقا في هذا الفصل حيث يتم تطبيق اللوغاريتم (حساب الطاقة اللوغاريتمي) على خرج المرشحات الترددية وذلك يؤدي إلى تسهيل الحصول استجابة المسار الصوتي كما سبق ووضحنا في التحليل الطيفي للمركبات حيث أن هذه الاستجابة هي التي تمثل شكل المقطع الصوتي (phoneme) المنطوق وبالتالي هذه الاستجابة هي ما يهمنا في الإشارة الناتجة حيث يعطى خرج هذه المرحلة بالمعادلة:

المعادلة (3-19):

$$|\log(s(f))| = |log(E(f).H(f))| = |log(E(f))| + |log(H(g))|$$

3-7-6-تطبيق التحويل التجيبي المتقطع Discrete Cosine Transform (DCT):

في هذه المرحلة يتم حساب المركبات التي تمثل كل إطار (cepstral coefficients) حيث أن هذه المركبات هي السمات المميزة للإشارة (بعد التجريب تم اختيار 13 مركبة لتمثيل كل إطار).

كما تعرفنا سابقا في التحليل الطيفي للمركبات أنه بتطبيق تحويل فورييه العكسي على لوغاريتم الإشارة فإن القسم الحقيقي من هذا التحويل يمثل المركبات الممثلة لهذه الإشارة.

إن تحويل DCT يمثل القسم الحقيقي من تحويل فورييه وبالتالي يمكن تطبيق هذا التحويل للحصول على هذه المركبات التي تمثل الإشارة.

4-الفصل الرابع: أدوات البحث

4-1-Google Colab:

تعد خدمة Google Colab من أبرز الخدمات التي تقدمها شركة Google لرواد مجالات الذكاء الاصطناعي والشبكات العصبونية، وعلماء تحليل البيانات، ومستخدمي لغة Python بشكل عام، حيث توفر محرر نصوص برمجية يعمل على متصفح الإنترنت، يتم تنفيذ النصوص البرمجية على سيرفرات شركة Google، ولعل أبرز استعمالات Google Colab هي تدريب واختبار نماذج الشبكات العصبونية لما توفره الخدمة من عتاد صلب يساعد أصحاب الأجهزة محدودة القدرات على تحقيق متطلبات قد لا تحققها حواسيبهم الشخصية، وتحوي مزايا مجانية ومزايا مدفوعة، استخدمنا في دراستنا النسخة المجانية من Google Colab.

4-1-1-أبرز القيود المفروضة من Google Colab:

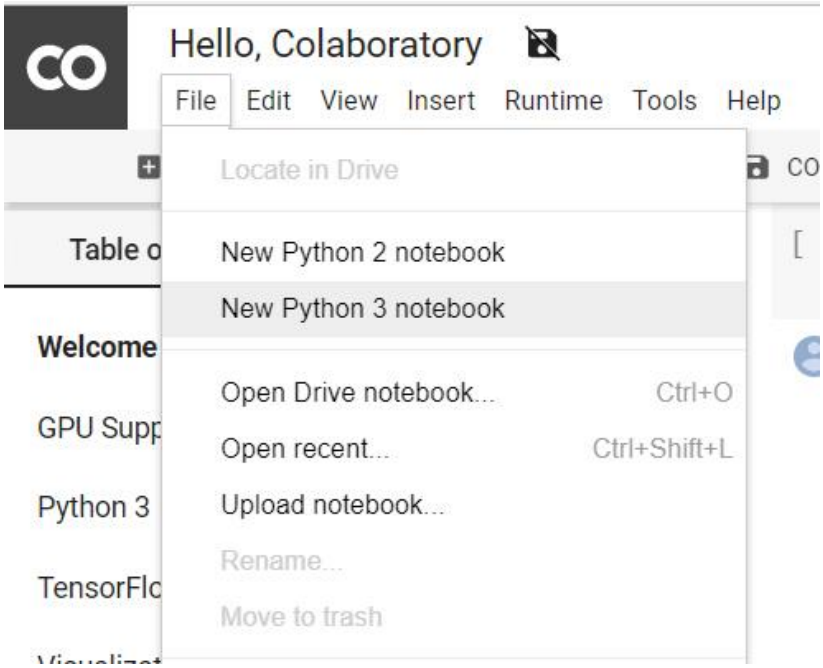
يمكن لمستخدم النسخة المجانية من Google Colab اختيار نوع وحدة المعالجة، إما وحدة معالجة مركزية CPU أو وحدة معالجة رسومية GPU وتوفر الخدمة مع كل وحدة معالجة مجموعة موارد من ذاكرة وصول عشوائية وقرص صلب بمساحة محددة، تختلف هذه الموارد حسب نوع وحدة المعالجة المختارة، من أجل وحدة معالجة مركزية CPU يتم توفير 12 GB من الذاكرة العشوائية و 107 GB مساحة قرص صلب، من أجل وحدة معالجة رسومية GPU يتم توفير 12 GB من الذاكرة العشوائية و 68 GB مساحة قرص صلب، يجدر بنا التنويه هنا أن في كل مرة يتصل فيها المستخدم بسيرفرات Google Colab يتم تخصيص وحدة معالجة عشوائية أي يمكن له اختيار GPU لكن نوع هذه الوحدة يتم اختياره بشكل عشوائي في كل محاولة اتصال جديدة، فقد يتم تخصيص وحدة GTX في إحدى المرات، وعند معاودة الاتصال ربما يتم تخصيص وحدة Tesla، يتم تحديد ذلك بشكل مجهول بالنسبة للمستخدم حسب الموارد المتاحة في اللحظة الحالية لدى سيرفرات الشركة، وتشير Google أن الموارد الأكثر كفاءة توفر عادة للمستخدمين الأقل استهلاكاً وقد تفرض الشركة قيوداً أحيانا على المستخدمين الذين يستعملون الموارد بشكل مفرط أو بشكل غير اقتصادي، كأن يستعمل المستخدم وحدة معالجة رسومية GPU في تنفيذ نصوص برمجية بسيطة لا تحتوي عمليات تدريب واختبار لشبكات عصبونية بشكل متكرر ولمدة طويلة، كما يمكن للمستخدم إشغال وحدتي معالجة كل منهما بعملية مختلفة، لكن عند تنفيذ نص برمجي وإغلاق المتصفح يستمر التنفيذ لمدة نصف ساعة أو أحيانا 45 دقيقة (حسب نوع وحدة المعالجة المستخدمة)، لكن عند عدم اتصال المستخدم بعد هذه المدة يتم إيقاف التنفيذ، وأقصى مدة تنفيذ يمكن الحصول عليها في حال عدم إيقاف الاتصال هي 12 ساعة متواصلة عند استخدام بعض وحدات المعالجة المتوفرة.

كل ما سبق يفرض علينا قيود محددة عند استخدام خدمة Google Colab، من حيث المدة الزمنية لعملية التدريب والموارد المتاحة، ويجب أخذ هذا الجانب بالحسبان عند اختيار حجم مجموعة البيانات المستخدمة وبنية الشبكة المصممة ودرجة تعقيدها، وإلا قد لا تستطيع خدمة Google Colab توفير الموارد اللازمة لنا.

4-1-2-التعرف على بيئة Google Colab:

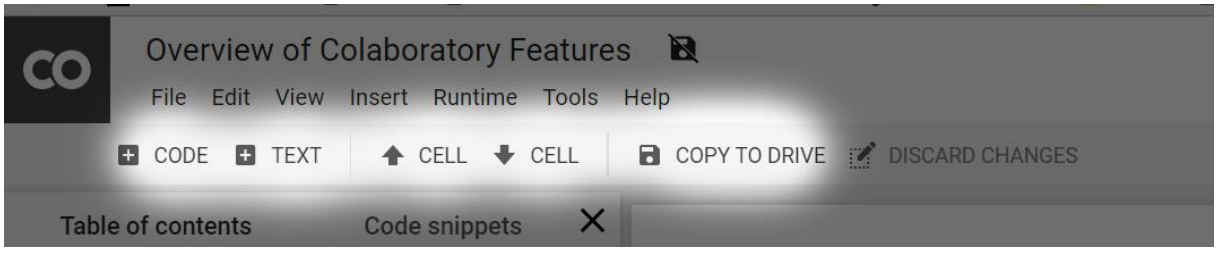
بمجرد الدخول إلى منصة Colab باستخدام حساب على Google يتم الانتقال فورا إلى منصة Jupyter جاهزة لكتابة النصوص البرمجية والتي تخزن تلقائيا في حساب Google Drive المرتبط بحساب Google للمستخدم، ومن ميزات Google Colab إمكانية مشاركة النصوص البرمجية مع مبرمجين آخرين ليتمكنوا من تعديلها أو حتى مشاهدتها فقط حسب تصريح المشاركة المحدد.

يمكن إنشاء Jupyter notebook لتحرير النصوص البرمجية باستخدام أحد إصدارات Python وهي إما Python2 أو Python3 وذلك كما يبين الشكل(4-1):



الشكل 4-1 إنشاء ملف Python في Colab

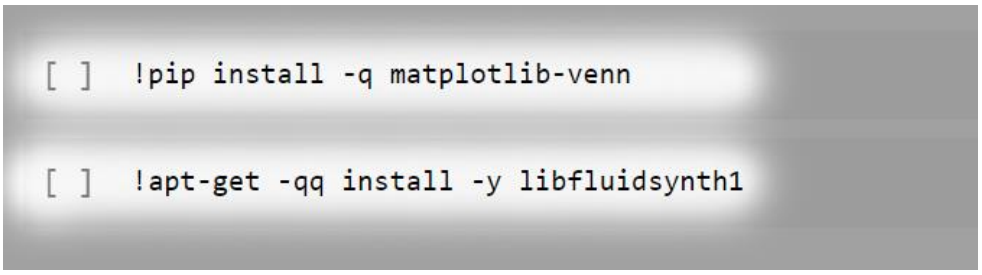
يمكن استخدام الاختصارات كما يوضح الشكل (4-2) لإنشاء خلية جديدة (Cell) من نوع Text أو Code أو التنقل بين الخلايا المختلفة أو حفظ الكود إلى حساب Google Drive، ويمكن تشغيل (Run) أي خلية بضغط Ctrl/CMD + Enter.



الشكل 4-2 الاختصارات في Colab

4-1-3-تحميل المكتبات ضمن Google Colab:

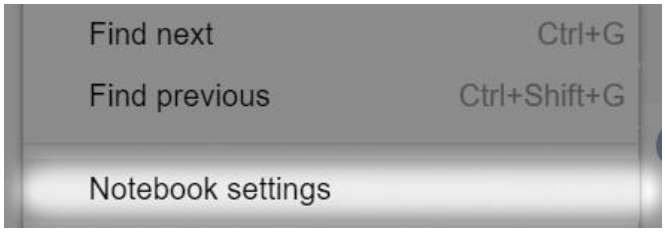
العديد من المكاتب الجاهزة للاستخدام تأتي مع Colab دون الحاجة لتحميلها مثل TenserFlow، ولكن يمكن تحميل أي مكتبة أخرى بطريقة بسيطة عبر إنشاء Code Cell جديدة، ويكتب فيها أمر التنصيب، كما يوضح الشكل (4-3) مثال تحميل المكتبة matplotlib-venn و libfluidsynth1 بطريقتين مختلفتين، الطريقة الأولى عبر استخدام التعليمة pip، والطريقة الثانية عبر استخدام التعليمة apt-get.



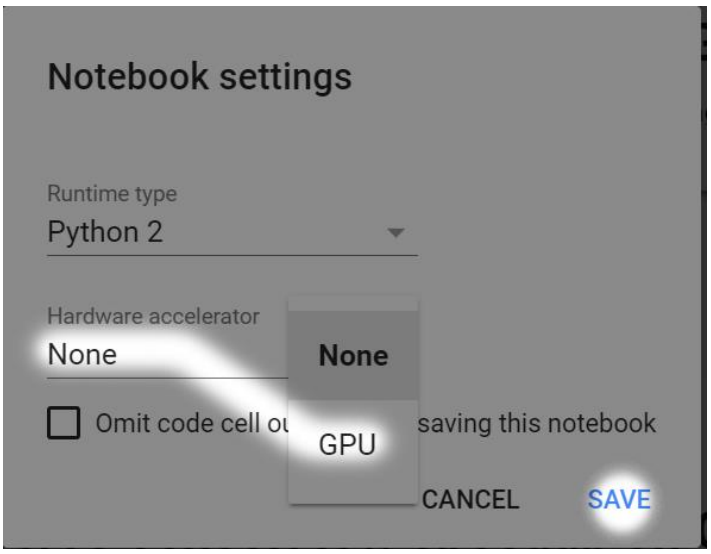
الشكل 4-3 تحميل مكتبات جديدة

4-1-4-استخدام الـ GPU ضمن Google Colab:

وهي الميزة الأهم التي تجعلنا نستخدم Google Colab حيث يوفر ميزة استخدام Tesla K80 GPU لمدة 12 ساعة متواصلة في المرة الواحدة ويمكن تحقيق ذلك كما يوضح الشكل (4-4) والشكل (4-5) من القائمة Edit.



الشكل 4-4 الخيارات في قائمة Edit



الشكل 4-5 استعمال ال GPU

4-2-لغة Python:

تعد لغة البرمجة Python من لغات البرمجة عالية المستوى التي تتميز بسهولة تعلمها وتعدد استخداماتها. ابتكرها وطورها جايدو ڤان أواخر ثمانينات القرن الماضي في مركز العلوم والحاسب الآلي بأمستردام، وتم الإعلان عنها لأول مره عام 1991، تم كتابة نواتها بلغة البرمجة سي وقد سميت Python نسبة إلي فرقه مسرحية ببريطانيا كانت تسمي مونتي بايثون.

لغة Python ابتكرت وطورت لتكون لغة عالية المقروئية، فهي تستخدم كلمات ومفاهيم إنجليزية شائعة الاستخدام بينما تستخدم العديد من اللغات البرمجية الأخرى علامات الترقيم، كما تستخدم Python المسافات البيضاء والإزاحة بدلا من الأقواس وذلك لكي يتم تحديد بداية ونهاية الدوال البرمجية.

تتميز بالعديد من الخصائص نذكر منها:

- سهولة الاستخدام والتعلم، حيث تمنح المبتدئ استمتاعًا غير مسبوقٍ خلال التعلم؛ إذ يصبح جاهدًا في التركيز على الحصول على الحلول عوضًا على صب الاهتمام على كيفية بناء الجملة.
- لغة مفتوحة المصدر ومجانية، يُسمح تداول لغة Python مجانًا وبكل حريةٍ دون أي قيودٍ، كما يُفتح الأفق أمام الراغبين في إجراء تعديلاتٍ على شيفرة المصدر بكل سهولةٍ، عالية المستوي فلا تحتاج لمراجعة التفاصيل.
- التحديث المستمر، إذ يحرص المبرمجون على إجراء التحديثات على الإصدارات بشكلٍ مستمرٍ وفي فتراتٍ متقاربة.
- قابلية النقل، يتمثل ذلك بإمكانية نقل البرامج المكتوبة بلغة Python بين المنصات المختلفة بسهولةٍ ودون أي تعقيداتٍ، ويُستدل من ذلك أنها قابلةٌ للتشغيل على جميع أنواع أنظمة التشغيل دون الحاجة لإجراء أي تغييراتٍ عليها إطلاقًا.
- الاندماج، تمتاز بإمكانية دمجها مع لغات البرمجة الأخرى للوصول إلى نتيجةٍ مرجوةٍ، ومن أهم هذه اللغات التي يمكن لها التضامن معها هي لغة C، C++ وغيرها، مما يجعل من الأداء مميزًا وعاليًا للغاية.
- لغة مفسرة تلقائيًا، إذ يتم تحويل أوامر لغة Python بشكلٍ تلقائيٍّ إلى لغةٍ يستوعبها الحاسوب وينفذ أوامرها على عكس اللغات الأخرى سواءً كانت ذات مستوى منخفضٍ أو عاليٍّ.
- لغة موجهة للكائنات، حيث تؤدي دورًا هامًا في تفسير المشكلات المعقدة بأسلوبٍ مميزٍ من خلال شطرها إلى أجزاءٍ أصغر فأصغر للتعامل معها.

4-2-1-مكاتب Python المستخدمة في مشروعنا:

- OS:

وهي من مكاتب Python الأساسية وسنستخدم منها توابع لقراءة المسارات Directories وإعادة تسمية الملفات فقط لا غير.

- Pandas:

مكتبة تحليل البيانات في Python وتحوي الكثير من التوابع للتعامل مع ملفات csv (Comma Separated Values) وتنظيم البيانات وسوف نستعملها في قراءة ملفات csv والتعامل مع السلاسل بنى المعطيات مثل أطر البيانات Data Frames والسلاسل Series.

- Numpy:

المكتبة المخصصة للتعامل مع المصفوفات في Python وتحوي العديد من التوابع الحسابية المفيدة كتابع القيمة المطلقة والقيمة المتوسطة وتابع حساب القيمة العظمى والدنيا ضمن مصفوفة وتوابع توليد الأرقام العشوائية، كما سنستخدم بعض توابعها لحساب تحويل فوريه.

- Matplotlib:

مكتبة خاصة برسم المخططات والقيم البيانية وسوف نستخدمها في استعراض بعض البيانات

- SciKit-Learn:

تحوي هذه المكتبة العديد من التوابع المستخدمة في مجال Machine Learning وكذلك Deep Learning، سنستخدم منها تابع لحساب دقة الاختبار وتقييم النتائج (accuracy_score) وتابع لحساب أوزان الشبكة بالاعتماد على توزع البيانات بين الأصناف compute_class_weights.

- tqdm:

مكتبة بسيطة تستخدم في تنفيذ الحلقات التكرارية ستساعدنا في متابعة تنفيذ الحلقات.

- pickle:

تستخدم هذه المكتبة في تخزين وقراءة الملفات الثنائية وسوف نستخدمها في تخزين النموذج وبارامترات التدريب (قيم الأوزان والانحياز)، بعد الانتهاء من التدريب لحفظ النتائج.

- Librosa:

المكتبة الكلاسيكية في Python للتعامل مع الإشارات الصوتية سنستخدم منها تابع Load لقراءة الإشارات الصوتية بتردد محدد، ويجب تحديد تردد التقطيع في هذا التابع لكي نتمكن من قراءة الإشارة.

- Scipy:

تحوي العديد من توابع معالجة الإشارة والتوابع الرياضية، لن نستخدمها في مشروعنا إلا لقراءة الإشارات الصوتية باستخدام التابع wavfile حيث يختلف هذا التابع عن تابع Load من Librosa، ولا يتطلب تحديد تردد التقطيع بل يقوم بإعادة التردد الأصلي للإشارة، وبذلك نستطيع معرفة تردد الإشارات الصوتية قبل إجراء عمليات تخفيض التردد وما شابهها.

- Python_speech_recognition:

تستخدم في التعامل مع تطبيقات معالجة الصوت، وسنستخدمها لحساب تحويل Mel-Filter Bank، و MFCC، حيث تحوي توابع جاهزة لذلك.

- Keras:

من أهم المكاتب في Python للتعامل مع الشبكات العصبونية، تعمل مع مكتبة Tensorflow الخاصة بالتعلم العميق والشبكات العصبونية أيضاً، وتحوي العديد من التوابع المهمة التي تسمح لنا ببناء طبقات الشبكة العصبونية وتدريبها وتحميل نماذج الشبكات المحفوظة سابقاً والعديد من العمليات على الشبكات العصبونية.

4-3-مجموعة البيانات Data Set:

أثناء بحثنا عن مجموعات بيانات يمكن استخدامها وجدنا بعض مجموعات البيانات الضخمة (10-20) GB قد لا تكون مناسبة للقيود التي تفرضها خدمة Google Colab في عملية التدريب، حاولنا استغلال الموارد المتاحة ووجدنا مجموعتي بيانات يمكن استخدامهما:

4-3-1-بيانات التدريب (Training Data Set):

بيانات التدريب التي استخدمناها هي مجموعة بيانات Data Set مجانية حصلنا عليها من الموقع OpenSLR (Open Speech and Language Resources) وهو موقع خاص بمجموعات البيانات الصوتية، واسم مجموعة البيانات هو "Free ST American Corpus" وتتضمن هذه البيانات تسجيلات صوتية لعشر أشخاص خمسة ذكور وخمس إناث، كل شخص لديه تقريبا 350 تسجيل صوتي، وجميع الأشخاص يتحدثون اللغة الإنجليزية، تتصف بيانات التدريب بأنه تم تسجيلها في بيئة مغلقة (غرفة) هادئة بواسطة هاتف ذكي، بتردد تقطيع 16 KHz، حجمها 529 MB، تحوي 3843 سجل، 2186 سجل للصنف Female، و1656 سجل للصنف Male.

4-3-2-بيانات الاختبار (Testing Data Set):

مجموعة بيانات Data set مجانية حصلنا عليها من موقع Kaggle، اسم مجموعة البيانات "Pygender Audio Set" وهي مجموعة تسجيلات صوتية عشوائية مأخوذة من مقاطع منتشرة عبر شبكة الانترنت،

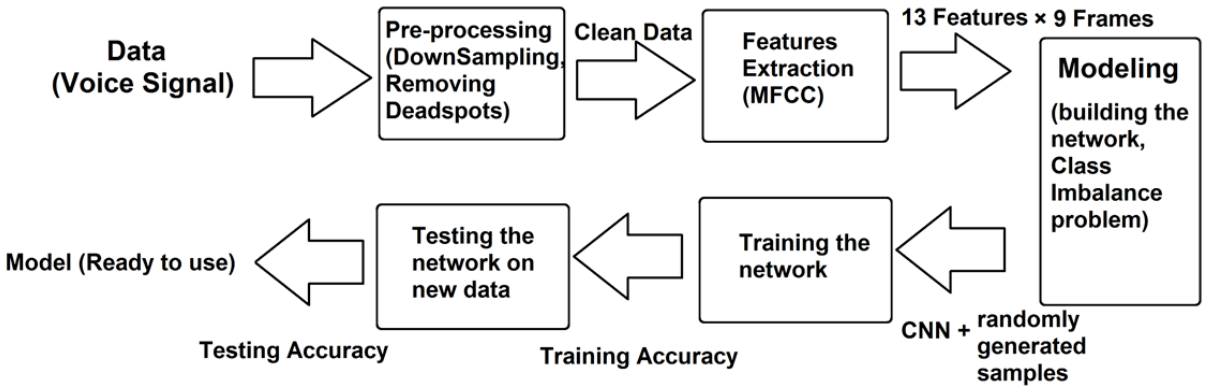
وبلغات متنوعة، وبتردد تقطيع 16 KHz، حجمها 335 MB، تحوي 1104 سجل، 558 سجل منها للصنف Female، و 546 سجل منها للصنف Male.

تتصف بأنها مسجلة بظروف عشوائية وبيئات مختلفة، تم تسجيلها بميكروفونات مختلفة، وبعضها تم تسجيله داخل غرفة مغلقة وبيئة هادئة والكثير منها تم تسجيله في بيئات مفتوحة أو مأخوذ من مقابلات ومقاطع تلفزيونية، والكثير منها يحوي ضجيج، وبعضها يحوي إشارات صوتية من أكثر من متكلم (هتافات، ضحك، أصوات جمهور)، سوف نستخدمها للاختبار.

بما أن مجموعة البيانات الأولى تحوي عدد سجلات أكبر، وبما أنها أكثر جودة من ناحية ظروف تسجيل الإشارات الصوتية، ارتأينا أن نستخدم مجموعات البيانات الأولى للتدريب، ومجموعات البيانات الثانية للاختبار، رغم اختلاف ظروف تسجيلها عن ظروف تسجيل بيانات التدريب، بهدف أخذ صورة عامة عن قدرة النموذج على التعميم.

يجدر بنا الذكر هنا أننا أنشأنا مجموعة بيانات خاصة بنا اختبرنا عليها النموذج في مرحلة لاحقة، سوف نتحدث عنها في نهاية الفصل الخامس.

5-الفصل الخامس: تحقيق النظام



الشكل 5-1 مراحل المشروع

5-1-النصوص البرمجية ومجلدات المشروع:

قبل البدء بمراحل تحقيق النظام يجدر بنا التنويه للنصوص البرمجية ومجلدات المشروع ومحتوى كل منها.

1. المجلدات:

- المجلد (__pycache__): يحوي ملفات يتم توليدها تلقائياً من محررات النصوص البرمجية الخاصة بلغة Python.
- المجلد (clean): يحوي ناتج عمليات المعالجة الأولية Pre-processing على مجموعة بيانات التدريب قبل إدخالها للنموذج.
- المجلد (models): يحتوي ملف ذو لاحقة model تم استخدامه لحفظ نواتج عملية التدريب.
- المجلد (pickles): يحتوي ملف ذو لاحقة p تم استخدامه لحفظ نواتج عملية التدريب.
- المجلد (Test_1_Results): يحوي بداخله مجلد آخر يتضمن سجلات مجموعة البيانات (Pygender AudioSet)، كما يحوي ملفين من الصيغة csv الأول test.csv يحوي أسماء السجلات والأصناف المقابلة لكل منها، والثاني predictions.csv يحوي أسماء السجلات والأصناف المقابلة لها ونتائج الاختبار بما فيها احتمال انتماء كل سجل لكل صنف من الأصناف، والتصنيف النهائي للنموذج لكل سجل من السجلات.
- المجلد (Test_2_Results): يحوي بداخله مجلد آخر يتضمن سجلات مجموعة البيانات الخاصة بنا والتي قمنا بإنشائها للاختبار عليها أيضاً (سوف نتحدث عنها في نهاية هذا الفصل)، كما يحوي ملفين من الصيغة csv الأول test.csv يحوي أسماء السجلات والأصناف المقابلة لكل منها، والثاني predictions.csv يحوي أسماء السجلات والأصناف المقابلة لها ونتائج الاختبار

بما فيها احتمال انتماء كل سجل لكل صنف من الأصناف، والتصنيف النهائي للنموذج لكل سجل من السجلات.

- المجلد (wavfiles): يحتوي على سجلات مجموعة بيانات التدريب (Free ST American Corpus)، قبل القيام بأي عمليات معالجة أولية عليها.

2. الملفات:

- الملف (GenderClassificationAudio.csv): يحوي أسماء سجلات مجموعة بيانات التدريب والصنف المقابل لكل سجل.
- النص البرمجي (cfg.py): يحوي صنف برمجي (Class) استخدمناه لتهيئة النموذج ببعض القيم.
- النص البرمجي (model.py): يحتوي على التعليمات البرمجية المسؤولة عن توليد العينات العشوائية من السجلات، وبناء الشبكة العصبونية، كما يحتوي التعليمات البرمجية المسؤولة عن مرحلة التدريب.
- النص البرمجي (plot_clean.py): يحتوي على التعليمات البرمجية المسؤولة عن رسم خرج مراحل خوارزمية MFCC، كذلك يحوي التعليمات البرمجية المسؤولة عن عمليات المعالجة الأولية للبيانات Pre-processing.
- النص البرمجي (predict.py): يحوي التعليمات البرمجية المسؤولة عن مرحلة الاختبار.
- النص البرمجي (test1_results.py): يحوي التعليمات البرمجية المسؤولة عن حساب نتائج الاختبار على مجموعة البيانات (Pygender Audioset).
- النص البرمجي (prepare_testing_data.py): يحتوي تعليمات برمجية تم استخدامها عن تحميل مجموعة البيانات (Pygender AudioSet) بهدف تنظيم سجلاتها وإعادة تسميتها وتنظيم ملف من الصيغة csv بأسماء تلك السجلات والأصناف المقابلة لكل منها.

5-2-تحميل بيانات التدريب:

استخدمنا خلال مشروعنا خدمة Google Colab المقدمة من شركة Google والتي تتضمن بيئة تطوير IDE تعمل على المتصفح وتسمح بتنفيذ برامج Python وتدريب الشبكات العصبونية باستخدام GPU مقدم من سيرفرات الشركة، ونظراً لأن رفع بيانات التدريب من أجهزتنا على Google Drive قد يستهلك وقتا طويلاً لذلك بداية قمنا بتحميل بيانات التدريب على Google Drive بشكل مباشر من الموقع الخاص بال Data setالى Drive مباشرةً باستخدام البرنامج التالي:

```
import os
import urllib.request
zip_url = http://www.openslr.org/resources/45/ST-AEDS-20180100_1-OS.tgz
urllib.request.urlretrieve(zip_url, 'SLR45.tgz')
```

ستقوم التعليمات السابقة بتحميل بيانات التدريب من الرابط وانشاء ملف مضغوط SLR45.tgz، لا يمكن فك ضغط هذا الملف ضمن Google Drive يدوياً، ولفك ضغط هذه الملفات نكتب التعليمات التالية:

```
import os
import sys
import math
import tarfile
dataset_path = "SLR45.tgz"
compressed_dataset_file_name = dataset_path
dataset_directory = "SLR45"
def extract_dataset(compressed_dataset_file_name, dataset_directory):
        tar = tarfile.open(compressed_dataset_file_name, "r:gz")
        tar.extractall(dataset_directory)
        tar.close()
        print("Files extraction was successfull ...")
extract_dataset(compressed_dataset_file_name, dataset_directory)
```

عرفنا تابع لفك الضغط ومررنا له اسم الملف المضغوط واسم المسار (المجلد SLR45) الذي نريد حفظ البيانات فيه، عند انتهاء تنفيذ البرنامج السابق نكون قد حملنا نسخة من بيانات التدريب على Google Drive ويمكننا البدء بمراحل المشروع الأساسية وتنفيذها على بيئة Google Colab.

يجدر بنا الذكر أن الفرق الأساسي بين كتابة النصوص البرمجية على الحاسب وعلى Google Colab هو مسارات المجلدات المستخدمة، لذلك سنعتمد في شرحنا على النصوص البرمجية القابلة للتنفيذ على الحاسب ويمكن استبدال المسارات بمسارات Google Drive عند تنفيذ النصوص البرمجية ذاتها على Google Colab.

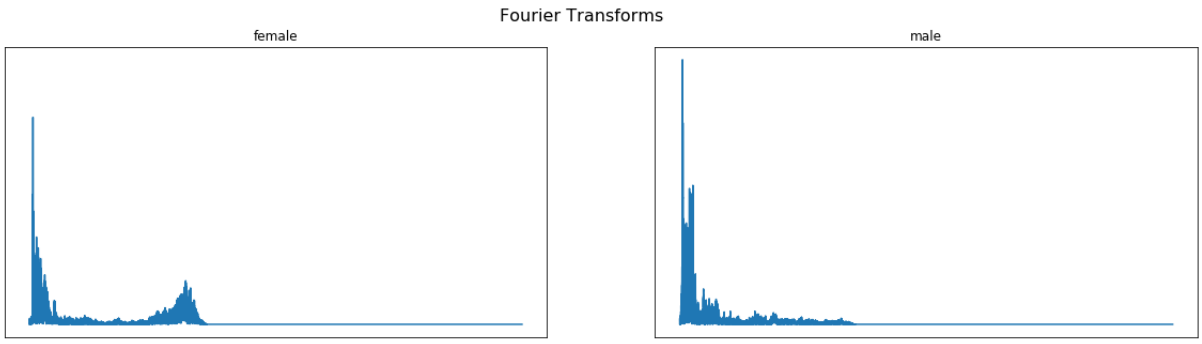
يجدر بنا التنويه أن ال Dataset الخاصة بعملية التدريب لدينا تحوي 3843 سجل.

5-3-تنظيف البيانات (Pre-processing):

تهدف هذه المرحلة إلى معالجة الإشارة الصوتية وإزالة الضجيج منها وتركيز المعلومات المفيدة، سنقوم بعمليتين أساسيتين في هذه المرحلة:

5-3-1-تخفيض تردد التقطيع (Down Sampling):

بما أننا سنستخدم خوارزمية MFCC لاحقاً في استخراج السمات (Features Extraction)، وكما رأينا أن هذه الخوارزمية تعتمد على تحليل تردد الإشارة، فقد وجد المختصون أنه في الإشارة الصوتية للإنسان تتركز المعلومات المهمة في الترددات المنخفضة، ونلاحظ هذا في المخطط التالي لتحويل فوريه:



الشكل 5-2 تركز المعلومات في الترددات المنخفضة من تحليل فوريه

وبالتالي لن نهتم هنا بالترددات المرتفعة مما يقلل من كمية المعلومات غير المفيدة بالنسبة لنا.

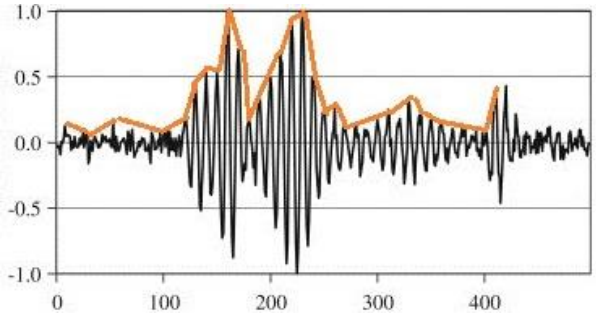
من الشائع في مثل هذه التطبيقات الخاصة بمعالجة صوت المتكلم أن يتم تخفيض تردد الإشارة إلى 16 KHz مما يسمح لنا بالاستفادة من أول 8 KHz (حسب شرط نايكوست)، يجدر بنا الذكر أن بعض التسجيلات الصوتية لدينا ترددها 16 KHz أصلاً لكن يجب تطبيق هذه الخطوة على جميع السجلات فهي خطوة أساسية ويجب التأكد من تنفيذها في حال وجود تسجيلات بتردد تقطيع مختلف، كما يجب تنفيذها على بيانات الاختبار عند وضع النموذج المصمم ضمن العمل قبل إدخالها لأننا دربنا نموذجنا على إشارات صوتية بتردد 16 KHz.

يمكن تنفيذ هذه الخطوة ببساطة باستخدام تابع load الموجود ضمن مكتبة Librosa والذي يسمح لنا بقراءة إشارة بتردد محدد، ومن ثم سنطبق على الإشارة ما تبقى من عمليات تنظيف ومعالجة أولية Pre-processing وفي النهاية سنقوم بإعادة كتابة الإشارة الصوتية في ملف wav. جديد.

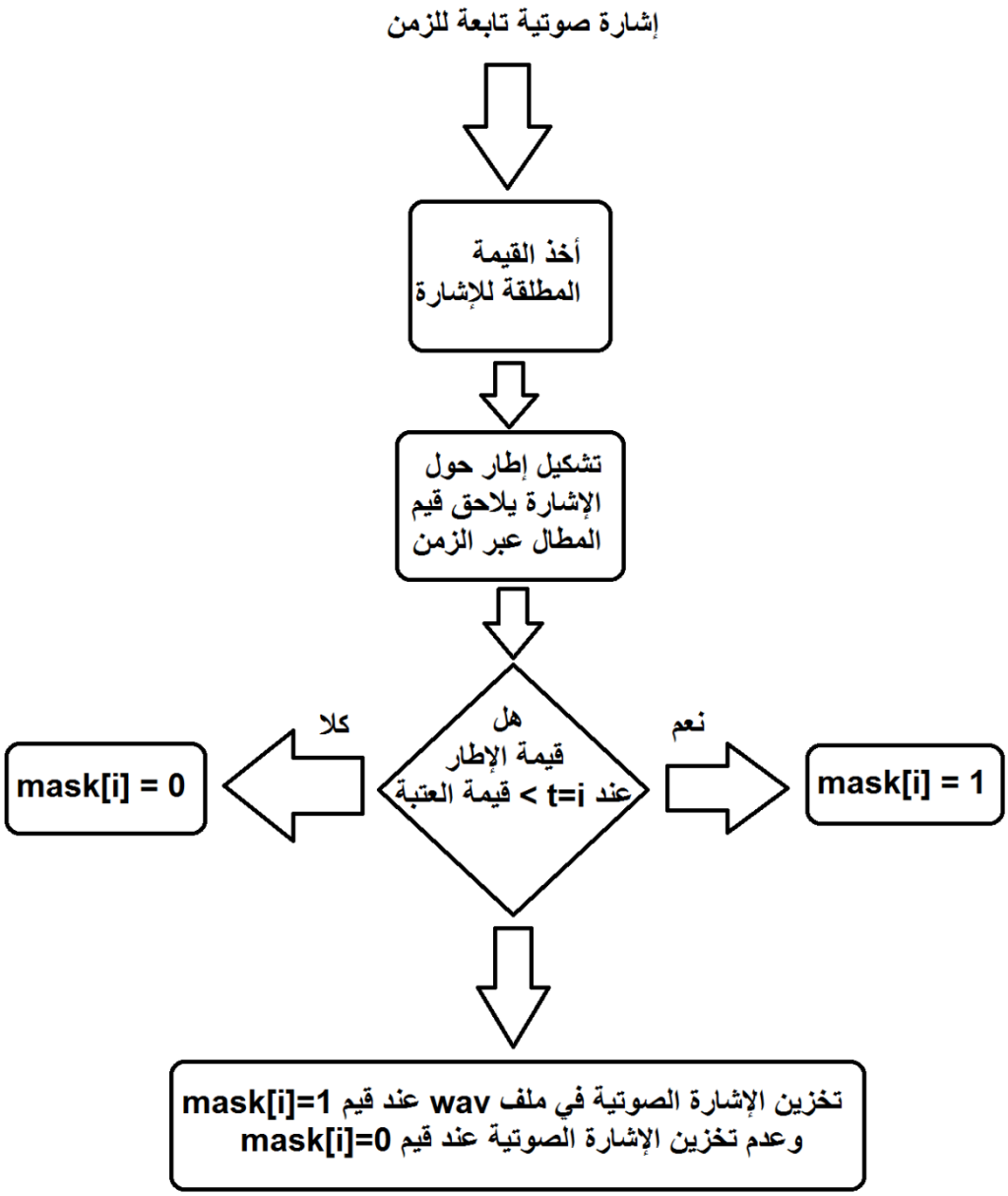
5-3-2-إزالة النقاط الميتة من الإشارة:

قد تحتوي بعض التسجيلات الصوتية لدينا على نقاط ميتة لا تحوي إشارة مفيدة، كمثال على ذلك لنفرض وجود سجل لدينا طوله 10 ثواني، أول ثانيتين منه لا تحوي إشارة صوتية للمتكلم، هنا يجب إزالة هاتين الثانيتين، أو أي إشارة مشابهة، لأنها تعتبر بمثابة إشارة ضجيج بالنسبة إلى الشبكة التي سوف نبنيها.

يمكن تطبيق ذلك باستخدام مفهوم Envelope of the signal وهو عبارة عن إطار يحيط بمطال الإشارة (اللون البرتقالي في الشكل التالي) تسمح لنا متابعته بمعرفة فيما لو كانت الإشارة تحوي صوت للمتكلم عند زمن معين أم لا، عن طريق مقارنة قيمة المطال بعتبة معينة.



الشكل 5-3 تابع Envelope لملاحقة قيم المطال



الشكل 5-4 آلية إزالة النقاط الميتة من السجلات

يمكن تطبيق المفهوم السابق مع خلال التابع التالي:

```
def envelope(y, rate, threshold):
    mask = []
    y = pd.Series(y).apply(np.abs)
    y_mean = y.rolling(window=int(rate/10), min_periods=1, center=True).mean()
    for mean in y_mean:
        if mean > threshold:
            mask.append(True)
        else:
            mask.append(False)
    return mask
```

قمنا بتحويل الإشارة إلى series أو سلسلة خاصة بمكتبة pandas، وأخذنا القيمة المطلقة للإشارة للمقارنة مع عتبة واحد موجبة للسهولة، ومن ثم قمنا بتطبيق تابع rolling الذي سوف ينشئ إطار حول الإشارة عن

طريق نافذة حجمها 1/10 من تردد التقطيع (ويمكن جعلها بحجم تردد التقطيع لكن ليس أكثر من ذلك) ومن ثم أخذ القيمة المتوسطة ضمن هذه النافذة.

ولكيلا تتأثر النافذة بالقيم قبل مبدأ الاحداثيات (عند x سالبة) عند بداية المسح والوقوف على مبدأ الإحداثيات جعلنا min_periods=1، والخاصية center = True.

بعد ذلك نمرر حلقة على قيم المطالات الخاصة بالإطار ونقوم بمقارنتها مع العتبة وحفظ القيمة True عندما تكون أكبر من العتبة (أي يوجد إشارة لصوت المتكلم هنا)، وحفظ القيمة False عندما تكون أصغر من العتبة، نحفظ هذه القيم في مصفوفة قناع mask، سوف تفيدنا هذه المصفوفة لاحقاً في حذف النقاط الميتة.

يمكن تطبيق العمليتين السابقتين (تخفيض تردد التقطيع وإزالة النقاط الميتة) معاً عن طريق حلقة تمر على سجلات التدريب لدينا كما يلي:

```
for f in tqdm(df.fname):
    signal, rate = librosa.load('wavfiles/'+f, sr=16000)
    mask = envelope(signal, rate, 0.0005)
    wavfile.write(filename='clean/'+f, rate=rate, data=signal[mask])
```

نلاحظ أننا حددنا تردد 16 KHz عن طريق تابع load أي قرأنا الإشارة بهذا التردد وسوف ننفذ بعدها التابع envelope لاستخراج القناع الخاص بكل إشارة، وجدنا بالتجريب أن أفضل عتبة للبيانات الخاصة لدينا هي 0.0005، ومن ثم سوف نكتب الإشارة في ملف wav. جديد بعد تخفيض التردد وعند قيم القناع mask وبالتالي نكون قد خفضنا التردد وأزلنا النقاط الميتة من الإشارة، قد يستغرق تنفيذ التعليمات السابقة وقتاً نظراً لأننا نقوم بمعالجة كل تسجيل صوتي وإعادة تخزينه، استغرق تنفيذ البرنامج السابق لدينا حوالي 30 دقيقة، والآن حصلنا على بيانات نظيفة يمكن استخراج السمات منها.

5-4-استخراج السمات (Features Extraction with MFCC):

كما شرحنا سابقاً MFCC هي خوارزمية لاستخراج السمات من البيانات الصوتية ولها ثلاث خطوات أساسية: تحويل فوريه، تحويل Mel-Filter Banks، تحويل التجيب المتقطع DCT.

تحوي مكتبة python_speech_features على مجموعة توابع تسمح لنا بإيجاد خرج تحويل MFCC وتحويل Mel-Filter Banks، أما تحويل فوريه فسنوجده حسابيا عن طريق توابع مكتبة Numpy.

سنستخدم التابع mfcc من مكتبة python_speech_features وسيعيد لنا لكل عينة دخل قيمة خرج أبعادها (9×13) حيث 13 هو عدد سمات MFCC الذي سوف نستخدمها، و9 هو عدد الإطارات الزمنية وهو متغير تابع لتردد التقطيع (16 KHz في دراستنا)، يتم تحقيق ذلك كما يلي:

```
def calc_fft(y, rate):
        تعريف تابع لحساب تحويل فوريه لاشارة y بتردد rate
    n= len(y)
        ايجار طول الإشارة للاستخدام في الحسابات
```

```
freq = np.fft.rfftfreq(n, d=1/rate)
```
إيجاد المركبة الترددية للإشارة وذلك بتمرير طول الإشارة ودورها
```
Y = abs(np.fft.rfft(y)/n)
```
إيجاد مطال الاشارة
```
return (Y, freq)
```
نعيد كل من المطال والمركبة الترددية حيث سوف نستخدمها في الرسم لاحقاً.

علما أننا سوف نطبق MFCC على كامل البيانات في مرحلة النمذجة Modeling بعد إيجاد عينات التدريب، لكن هنا سوف نرسم خرج مراحل MFCC ونطبقه على سجل واحد من كل صنف لكي نرى الخرج.

```
for c in classes:
    wav_file = df[df.label == c].iloc[0,0]
```
قراءة اسم الملف عد الموقع 0،0 أي قراءة اسم أول ملف من كل صنف
```
    signal, rate = librosa.load('wavfiles/'+wav_file, sr=16000)
```
قراءة الإشارة الصوتية حسب اسم الملف المقروء بتردد 16 KHz
```
    mask = envelope(signal, rate, 0.0005)
    signal = signal[mask]
```
إزالة النقاط الميتة كما سبق
```
    signals[c] = signal
    fft[c] = calc_fft(signal, rate)
```
حساب تحويل فوريه وفق للتابع الذي شرحناه سابقاً وتخزينه لكل صنف.
```
    bank = logfbank(signal[:rate], rate, nfilt=26, nfft=400).T
```
إيجاد تحويل Mel-Filter Banks ونذكر هنا ان البارامتر nfft يحسب من العلاقة:

$$N\ fft = Sampling\ Rate \times Windows\ Size$$

أي حاصل جداء تردد التقطيع في حجم نافذة التقطيع (انظر لفقرة النوفذة سابقاً)، أي:

$$N\ fft = 16\ KHz \times 25\ ms = 400$$

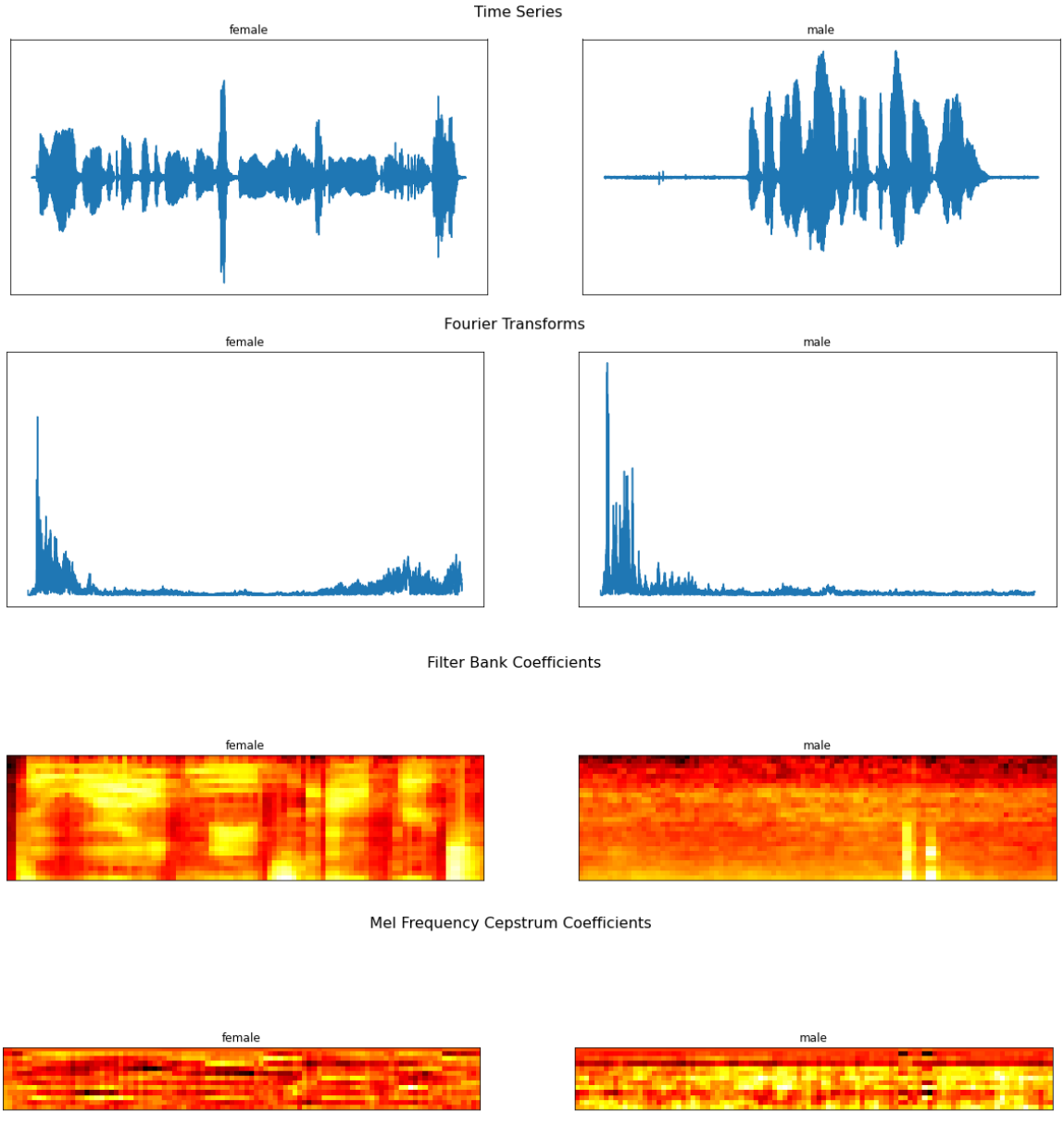
وهنا جرت العادة على اختيار حجم نافذة 25 ms في تطبيقات التعرف على الصوت البشري ومعالجة الصوت بشكل عام، كما يؤخذ عدد الفلاتر nfilt=26.

نعود لبقية الكود:
```
    fbank[c] = bank
```
تخزين نتائج تحويل Mel-Filter Banks لكل صنف
```
    mel = mfcc(signal[:rate], rate, numcep=13, nfilt=26, nfft=400).T
    mfccs[c] = mel
```
بشكل مشابه لتحويل Mel-Filter Banks نوجد خرج تحويل MFCC لكل صنف ونخزنه.

الآن نرسم النتائج (علماً أن التوابع التالية توابع رسم باستخدام تعليمات Matplotlib بسيطة يمكن الاطلاع عليها):
```
plot_signals(signals)
```

```
plt.show()
plot_fft(fft)
plt.show()
plot_fbank(fbank)
plt.show()
plot_mfccs(mfccs)
plt.show()
```

ويكون الخرج:



الشكل 5-5 خرج مراحل MFCC الأساسية

- نلاحظ في المجال الزمني عدم القدرة على تحديد جنس المتكلم وعدم وجود علامات مميزة بين الصنفين.

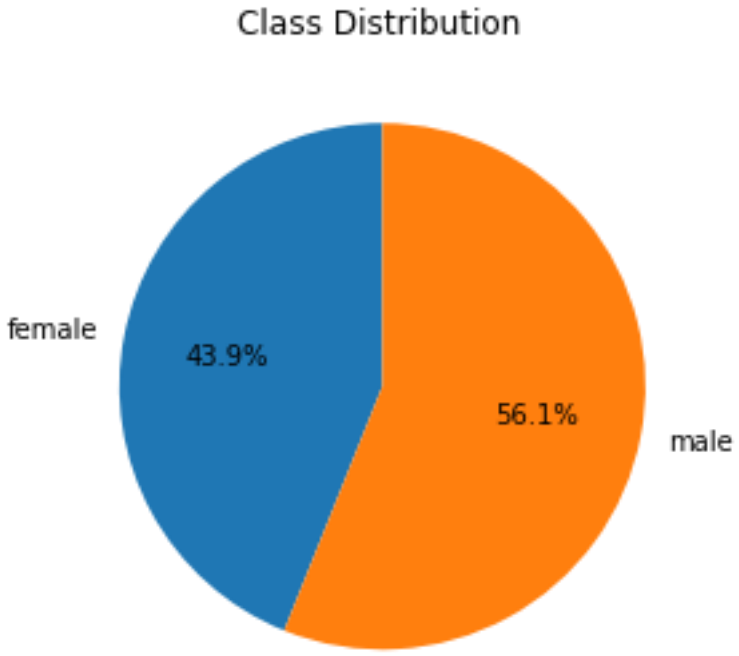
- بعد تنفيذ تحويل فورييه نبدأ بملاحظة الفرق حيث أن تتركز المعلومات في صوت الذكر عند الترددات الأدنى مقارنة بصوت الأنثى.
- عند إجراء تحويل Mel-Filter Banks نجد أن المقارنة أصبحت أوضح.
- بإتمام الخطوة الأخيرة من MFCC وإجراء تحويل التجيب المتقطع DCT نجد أنه أصبح بالإمكان تمييز الفرق بسهولة أكثر.

5-5-النمذجة (Modeling):

الآن بعدما أصبح بإمكاننا استخراج السمات من الإشارات الصوتية ننتقل لمرحلة النمذجة وبناء الشبكة.

5-5-1-توليد العينات العشوائية ومشكلة عدم توزع البيانات وفق الأصناف:

قبل بناء الشبكة نلاحظ وجود عدم توازن بين كمية العينات المتوفرة للصنفين، فإذا رسمنا مخطط بياني يعبر عن توزع البيانات بين الصنفين (بعد مرحلة تنظيف البيانات طبعاً) نجد ما يلي:



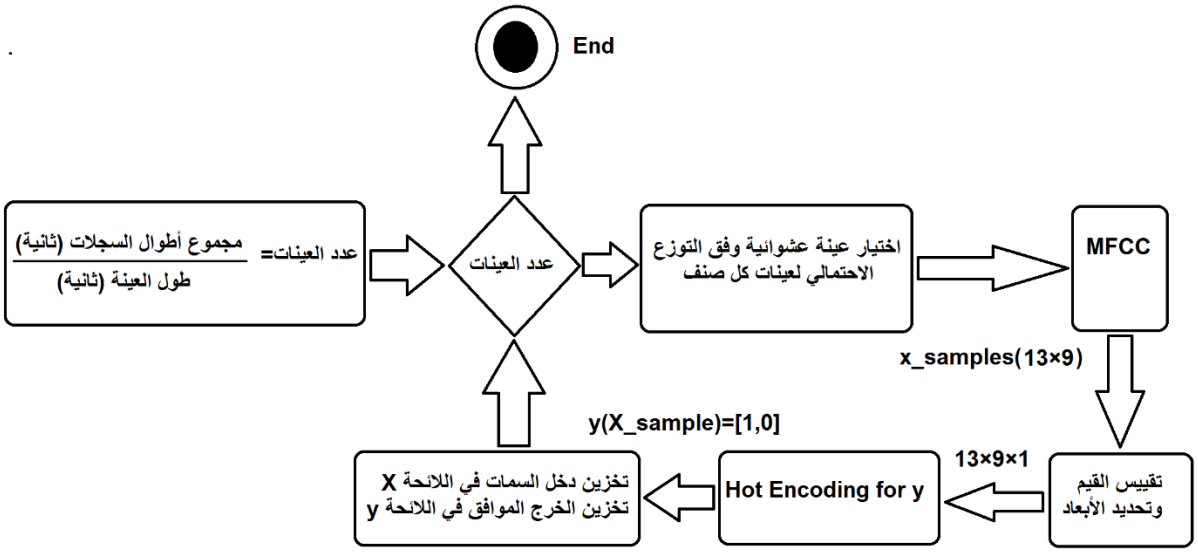
الشكل 5-6 نسبة توزع البيانات بين الصنفين

نلاحظ أن بيانات الصنف Male أكثر من بيانات الصنف Female، وهذه مشكلة يجب تحسينها، لأننا لو دربنا الشبكة على البيانات كما هي الآن فسوف تتعلم على التقاط عينات من الصنف Male أكثر من الصنف Female مما يوثر سلباً على النتائج. وتدعى هذه الحالة Class Imbalance.

وان تخلصنا من قسم من بيانات الصنف الأول Male لإحداث توازن بين بيانات الصنفين نكون قد خسرنا بيانات قيّمة، قد يكون الحل أحيانا الحصول على المزيد من بيانات الصنف Female لكن هذا الحل

غير عملي وغير قابل للتطبيق في كثير من الأحيان، أحيانا تكون هذه هي البيانات المتوفرة فقط ويجب علينا استغلالها بالشكل الأمثل.

سنحل المشكلة هنا بتقسيم السجلات الصوتية لدينا لعينات صغيرة (طول كل منها 1/10 من الثانية)، والتدريب عليها عشوائياً ولكن باحتمال اختيار عشوائي مساوٍ لنسبة توزع البيانات بين الصنفين، وبذلك نكون قد دربنا على أقصى كمية ممكنة من البيانات المتوفرة وبتوازن بين الصنفين دونما أن تتعلم الشبكة من صنف أكثر من الآخر، كما أنه لتقسيم البيانات لعينات صغيرة محاسن أخرى، أهمها أننا سوف نعلم الشبكة على التصنيف باستخدام مقطع صوتي طوله 1/10 من الثانية، يعد هذا تحدٍ لكن إن حققنا نتائج جيدة ربما يتمكن نموذجنا في المستقبل مع بعض التحسينات أن يقوم بتصنيف الإشارات الصوتية بنظام تصنيف حي (Live Audio Classification) أو ربما ضمن الزمن الحقيقي (Real-time system).



الشكل 5-7 بناء العينات

إذن بداية سوف نوجد عدد العينات التي سوف ندرب عليها كما يلي:

```
df = pd.read_csv('GenderClassificationAudio.csv')
df.set_index('fname', inplace=True)
```

نقرأ أسماء السجلات من ملف csv. الذي أنشأناه سابقاً ونحدد الindex وفق عمود الاسم fname.

```
for f in df.index:
    rate, signal = wavfile.read('clean/'+f)
    df.at[f, 'length'] = signal.shape[0]/rate
```

نقوم بإنشاء عمود جديد يحوي طول كل تسجيل صوتي لدينا بالثواني.

```
classes = list(np.unique(df.label))
```

نقرأ الأصناف الموجودة لدينا (Female،Male) ونخزنها في list.

```
class_dist = df.groupby(['label'])['length'].mean()
```

نوجد سلسة فيها عنصرين كل عنصر يعبر عن نسبة توزع بيانات صنف (56.1% للصنف الأول و43.9% للصنف الثاني كما في الشكل السابق).

```
n_samples = 2 * int(df['length'].sum()/0.1)
```

هنا نقوم بحساب عدد العينات الناتجة عن تقسيم السجلات الصوتية لمقاطع طول كل منها 0.1 من الثانية وسنضرب الناتج ب 2 للتأكد من الحصول على عينات كافية واستغلال كافة البيانات الممكنة، نتج معنا عدد عينات 202380 عينة (سنستخدم الناتج في حلقة لاستخراج العينات لاحقاً).

```
prob_dist = class_dist/class_dist.sum()
```

نقوم بتقييس توزع بيانات الصنفين لتصبح نسبة بين 0 و1.

```
choices = np.random.choice(class_dist.index, p=prob_dist)
```

التعليمة السابقة تقوم بتوليد اختيار عشوائي بين الأصناف الموجودة اعتماداً على نسبة توزع البيانات، وسنستخدم هذا الأسلوب لتجهيز العينات، الآن نقوم بكتابة تابع نهائي لتحضير العينات العشوائية لعملية التدريب واستخراج السمات منها كما يلي:

```
def build_rand_feat():
    X = []
    y = []
```

تعريف list خاصة بالدخل وlist خاصة بالخرج سنخزن فيهما العينات

```
    _min, _max = float('inf'), -float('inf')
```

تعريف القيم min وmax وتهيئة كل منهما بقيم لا نهائية (سنستخدمهما لتقييس النتائج لاحقاً)

```
    for _ in tqdm(range(n_samples)):
```

حلقة تنفذ n مرة حيث n هو عدد العينات الذي قمنا بحسابه سابقاً

```
        rand_class = np.random.choice(class_dist.index, p=prob_dist)
```

عملية اختيار صنف عشوائي وفق توزع البيانات كما شرحنا سابقاً

```
        file = np.random.choice(df[df.label==rand_class].index)
        rate, wav = wavfile.read('clean/'+file)
        label = df.at[file, 'label']
```

اختيار اسم ملف عشوائي لتسجيل صوتي موافق للصنف المختار وقراءته وتخزين الصنف الموافق له

```
        rand_index = np.random.randint(0, wav.shape[0]-config.step)
        sample = wav[rand_index:rand_index+config.step]
```

اختيار موقع عشوائي ضمن التسجيل الصوتي المختار وتحديد عينة ضمنه حجمها يساوي حجم الخطوة 0.1 من الثانية كما اتفقنا.

```
        X_sample = mfcc(sample, rate,
                        numcep=config.nfeat, nfilt=config.nfilt, nfft=config.nfft)
```

استخراج السمات وفق MFCC من العينة المختارة

```
        _min = min(np.amin(X_sample), _min)
        _max = max(np.amax(X_sample), _max)
```

تحديث قيم min وmax في حال ورود قيم أحدث

```
        X.append(X_sample)
        y.append(classes.index(label))
```

تخزين النتائج (السمات المستخرجة من العينة العشوائية في x والصنف المقابل لها فيy)

```
X, y = np.array(X), np.array(y)
X = (X - _min)/(_max - _min)
```

بعد الانتهاء من تنفيذ الحلقة سنقوم بتحويل كل من x و y لمصفوفات numpy لنجري عملية التقييس اللازمة

```
X = X.reshape(X.shape[0], X.shape[1], X.shape[2], 1)
```

هنا نقوم بإضافة بعد إضافي لأبعاد الدخل وهذه العملية خاصة ببنية الشبكة CNN التي سوف ندرب عليها.

```
y = to_categorical(y, num_classes=2)
```

يقوم تابع to_categorical من مكتبة Keras بعملية تسمى Hot Encoding وهي هنا تحويل الأصناف (Male،Female) ذوي الترميز (0، 1) إلى ترميز ثنائي بحيث لو فرضنا أن سجل ما من الصنف Male ستكون قيمة y الخاصة به بعد ترميز Hot Encoding هي مصفوفة ثنائية (لأنه لدي صنفين هنا) قيمتها y =[1,0] وكذلك في حال كان لدينا سجل ينتمي للصنف Female سيكون y الخاص به y=[0,1]

وهذه العملية ضرورية عند التعامل مع Keras و Tensorflow في عملية التدريب، كما أنه من الضروري جعل الخرج مكون من مركبتين لأنه في عملية الاختبار سيكون الخرج مكون من قيمتين مجموعهما 1، مثال y = [0.85 , 0.15] أي الشبكة تتوقع في هذه الحالة أن احتمال انتماء الدخل للصنف الأول 85% واحتمال انتماء الدخل للصنف الثاني 15% أو يمكن القول أنه تم تصنيف 15% من الإشارة الصوتية الواردة للصنف الثاني وتم تصنيف 85% للصنف الأول وهكذا.

```
return X, y
```

هنا نقوم بإعادة كل من مصفوفة السمات المستخرجة X ومصفوفة الأصناف المقابلة لها y.

سوف يعيد التابع السابق قيم العينات العشوائية اللازمة للتدريب، بما يتناسب مع توزع بيانات الصنفين وبالتالي سوف يحل مشكلة Class Imbalance.

5-5-2-بناء الشبكة:

سوف نستخدم شبكة عصبونية التفافية CNN، كما شرحنا سابقاً المكونات الأساسية للشبكات العصبونية الالتفافية هي:

- الطبقة الالتفافية Convolutional Layer
- طبقة التجميع Pooling Layer
- طبقة الارتباط الكامل Fully-connected Layer

عند اختيار بنية الشبكة العصبونية المناسبة يجب البحث عن نماذج مختلفة والاعتماد على التجريب، وجدنا عدة نماذج لشبكات تستخدم في حالة التدريب على الإشارات الصوتية ووجدنا أنسبها الشبكة التالية:

- الطبقة الالتفافية Convolutional Layer:

سنستخدم أربع طبقات التفافية بمرشحات من الحجم (3،3) عددها متزايد في كل طبقة من 16 وصولا إلى 128، حيث تتعلم كل طبقة سمات أكثر تعقيداً ناتجة عن خرج الطبقة السابقة، مع الحفاظ على أبعاد الدخل بجعل بارامتر padding = 'same'، تزداد قدرة الشبكة على التعلم بزيادة عدد الطبقات الالتفافية ونحن نحتاج نموذج قادر على تعلم سمات كافية لكن غير معقد لدرجة حصول حالة Overfitting، وجدنا أن أغلب الشبكات المستخدمة لدراسة الإشارة الصوتية للمتكلم البشري تحوي بين 3 و 5 طبقات التفافية، فاعتمدنا أن أربع طبقات كافية للحالة المدروسة.

- دالة التفعيل Activation function:

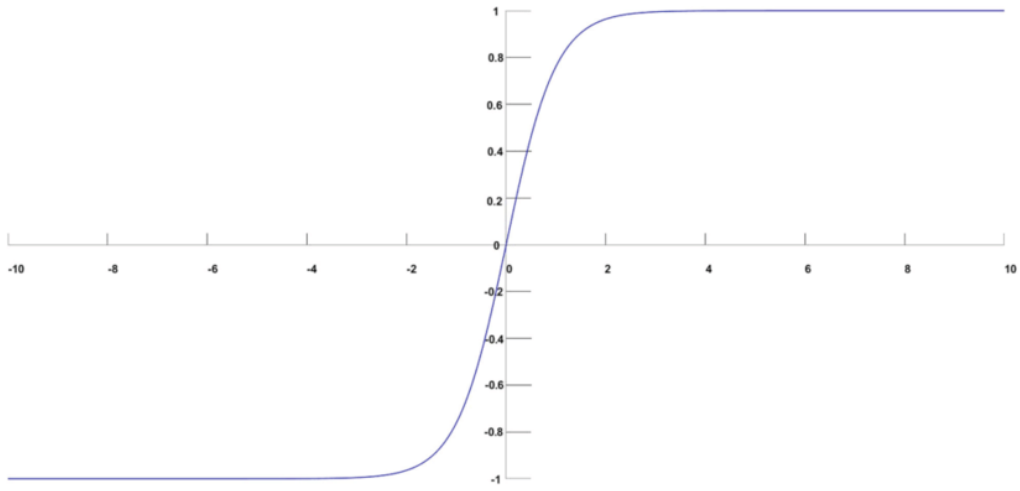
أغلب النماذج التي وجدناها تستخدم دالة تفعيل (ReLU) وسنعتمد دالة تفعيل Rectangular Linear Unit (ReLU).

- طبقة التجميع Pooling Layer:

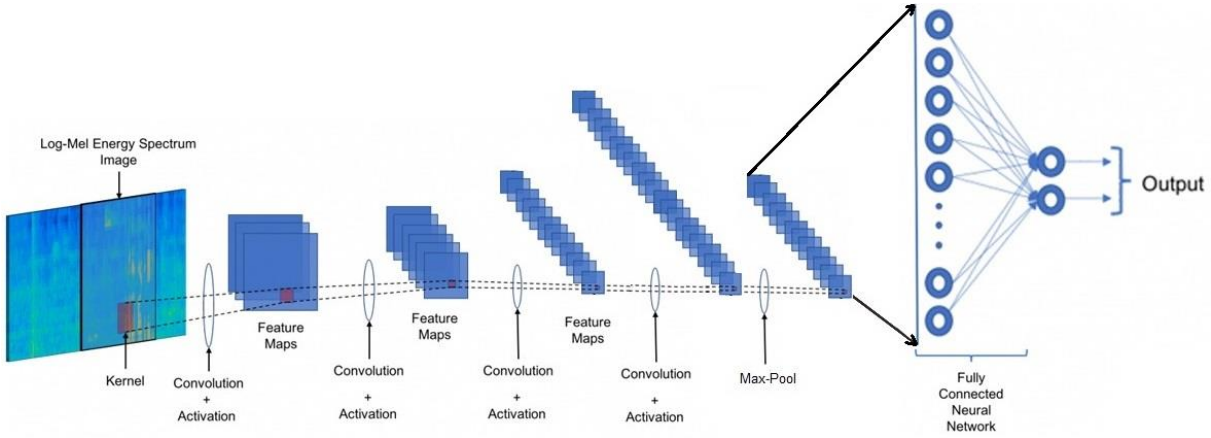
يلزمنا الآن التفكير بطبقات التجميع Pool، تقوم طبقة التجميع بتخفيض أبعاد الدخل الوارد إليها عادة من طبقة التفافية، وبما أن أبعاد الدخل لدينا هي (1،9،13) حيث 13 هو عدد معاملات MFCC الذي حددناه و 9 مركبة زمنية تعبر عن عدد الإطارات الزمنية قيمتها ناتجة عن قيمة تردد التقطيع وتتغير بتغير تردد الإشارة الصوتية، والمركبة الثالثة قمنا بإضافتها لمراعاة شروط أبعاد الدخل اللازمة لشبكات CNN، نلاحظ أن الأبعاد ليست كبيرة وبالتالي لا نحتاج لكثير من طبقات التجميع Pool، اخترنا في نموذجنا وضع طبقة من النوع MaxPool وحيدة من الحجم (2,2) ستقوم بتخفيض حجم الأبعاد إلى النصف مما يخفض كمية الحسابات ويجنبنا حالة overfitting، في حال كانت أبعاد الخرج كبيرة يجب إضافة المزيد من طبقات التجميع لكن في حالتنا طبقة واحدة ستكون كافية.

- طبقات الارتباط الكامل Fully-connected Layer:

في البداية سنضع طبقة Dropout بنسبة 0.5 أي خلال كل دورة تدريب احتمال إطفاء كل عصبون في الشبكة هو 0.5 لمنع حصول overfitting، كما سنضيف بالتأكيد طبقة flatten لتسطيح الخرج وتحويله من ثلاثي البعد إلى شعاع أحادي البعد، وسنضع ثلاث طبقا Dense بأحجام متناقصة من 128 بداية وصولاً إلى خرج نهائي بعدد الأصناف (2) حيث تابع تفعيل الطبقة النهائية سيكون Softmax الموضح في الشكل (5-8) وهو التابع المستخدم في طبقات الخرج في حالات التصنيف ويكون دخله احتمال انتماء عينة للأصناف وخرجه تصنيف هذه العينة.



الشكل 5-8 تابع التفعيل Softmax



الشكل 5-9 بنية الشبكة

اعتمدنا النموذج السابق بعد البحث والتجريب حيث وجدناه مناسب لحالة دراستنا، والمعيار الأساسي هو النتائج.

نقوم بتعريف تابع لبناء الشبكة:

```
def get_conv_model():
    model = Sequential()
    model.add(Conv2D(16, (3,3), activation='relu', strides=(1, 1),
                     padding='same', input_shape=input_shape))
    model.add(Conv2D(32, (3,3), activation='relu', strides=(1, 1),
                     padding='same'))
    model.add(Conv2D(64, (3,3), activation='relu', strides=(1, 1),
                     padding='same'))
    model.add(Conv2D(128, (3,3), activation='relu', strides=(1, 1),
                     padding='same'))
    model.add(MaxPool2D((2,2)))
    model.add(Dropout(0.5))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(2, activation='softmax'))
```

```
    model.summary()
    model.compile(loss='categorical_crossentropy',
            optimizer='adam',
            metrics=['acc'])
    return model
```

- تابع حساب الخطأ المتبع هو 'categorical_crossentropy' وهو الأنسب والمستخدم عادة في مسائل التصنيف.
- وخوارزمية التدريب المُتبعة هي: Adam (تم شرحها في فقرة الانحدار التدريجي سابقاً)
- معيار التقييم (Accuracy):

سوف يتم حساب الدقة وفق القانون:

$$Accuracy= ((TP + TN) / (TP + TN + FP + FN))$$

(True Positive: TP): القيمة التنبئية الإيجابية الصحيحة؛ عدد التنبؤات التي نجح فيها النموذج في تعرف الأنماط الصحيحة، مثلاً الشخص مريض والنموذج تنبأ بذلك.

(False Positive): القيمة التنبئية الإيجابية الخاطئة؛ عدد التنبؤات التي فشل فيها النموذج في تعرف الأنماط الصحيحة، مثلاً الشخص مريض والنموذج تنبأ بأنه غير مريض.

(True Negative): القيمة التنبئية السلبية الصحيحة؛ عدد التنبؤات التي نجح فيها النموذج في تعرف الأنماط الخاطئة، مثلاً المريض غير مريض والنموذج تنبأ بذلك.

(False Negative): القيمة التنبئية السلبية الخاطئة؛ عدد التنبؤات التي فشل فيها النموذج في تعرف الأنماط الخاطئة، مثلاً المريض غير مريض والنموذج تنبأ بأنه مريض.

5-6-تدريب الشبكة:

بعد أن قمنا ببناء الشبكة وتحديد بنيتها، وتجهيز عينات التدريب، سنقوم بتدريب الشبكة، حيث قمنا بالتدريب على سيرفرات Google Colab كما ذكرنا سابقاً باستخدام GPU ويمكن التحقق من نوع الGPU المستخدم ضمن Colab عن طريق التعليمة التالية:

```
!nvidia-smi --query-gpu=gpu_name,Driver_version,memory.total --format=csv
```

وكان خرج التعليمة:

```
[ ]   1 !nvidia-smi --query-gpu=gpu_name,driver_version,memory.total --format=csv

      name, driver_version, memory.total [MiB]
      Tesla K80, 418.67, 11441 MiB
```

الشكل 5-10 التحقق من نوع ال GPU المستخدم ضمن بيئة Google Colab

أي أن نوع ال GPU الذي وفره لنا Google Colab هو Tesla K80.

للقيام بعملية التدريب نكتب التعليمات التالية:

```
X, y = build_rand_feat()
```
استدعاء التابع الذي ناقشناه سابقاً المسؤول عن توليد العينات العشوائية اعتماداً على توزع الأصناف
```
y_flat = np.argmax(y, axis=1)
```
هنا نقوم بعكس عملية Hot Encoding وتحويل المصفوفة y لشعاع أبعاده n×1 حيث n هو عدد العينات وتخزين الناتج في y_flat سنستخدم y_flat في حساب مصفوفة أوزان class_weight لاحقاً تفيد في التأكد مجدداً من تحسين حالة Class Imbalance التي كانت موجودة سابقاً أما y سنستخدمها في عملية التدريب.
```
input_shape = (X.shape[1], X.shape[2], 1)
```
نحدد أبعاد الدخل مع ملاحظة البعد الثالث الضروري وفقاً لبنية شبكات CNN
```
model = get_conv_model()
```
نستدعي الشبكة التي صممناها سابقاً ليتم بناءها
```
class_weight = compute_class_weight('balanced',np.unique(y_flat),y_flat)
```
هنا نستخدم التابع compute_class_weight من مكتبة sklearn كما ذكرنا سابقاً بسبب وجود حالة Class Imbalance لدينا، رغم أننا عالجنا المشكلة سابقا بتابع build_rand_feat لكن استخدام التابع compute_class_weight قد يساهم في توفير خطوة إضافية لتحسين توازن الأصناف ويساعد في رفع دقة النموذج ولو قليلاً كما وجدنا عند البحث.
```
checkpoint=ModelCheckpoint(config.model_path, monitor='val_acc', verbose=1, mode='max',
save_best_only=True, save_weights_only=False, period=1)
```
التعليمة السابقة تساعدنا في حفظ النموذج بعد كل تكرار epoch في حال تحسنت الدقة ويجدر بنا الذكر هنا أن حفظ النموذج ليس جزء من عملية التدريب لكن كون تدريب الشبكات العصبونية قد يأخذ وقتاً طويلا ويستهلك الكثير من الموارد فمن الأفضل دائماً حفظ النتائج لاختبارها لاحقا دون الحاجة لإعادة التدريب.
```
model.fit(X,y,epochs=10,batch_size=32,shuffle=True,validation_split=0.1,callbacks=[checkp
oint])
```
العملية السابقة تعلن عن بداية التدريب بتمرير العينات X والأصناف المقابلة لها y، كما قمنا بتحديد عدد دورات التدريب epochs = 10 أي سنقوم بالتدريب على البيانات 10 مرات وفي كل مرة سنتحقق من زيادة دقة التدريب، قد يكون هذا الرقم أكبر أو أصغر من اللازم وسوف نرى ذلك من خلال النتائج وتقييمها.

قمنا بتحديد الحجم الافتراضي للدفعات batch_size = 32، طريقة تدريب شبكتنا هي بإشراف مُعلِّم (Supervised) ومن نوع (Batch training)؛ أي أن تعديل الأوزان يتمّ بعد إدخال دفعة batch من 32 عينة هنا، على عكس النوع المُقابل (Online training) الذي يتم فيه تعديل الأوزان بعد إدخال كُل البيانات دفعة واحدة، والتي قد تكون مكلفة جداً في حال البيانات الضخمة مثل مشروعنا (3800 سجل تدريب).

كما قمنا بتفعيل خاصية الخلط shuffle=True رغم أننا نقوم بانتقاء العينات عشوائياً وفق التابع build_rand_feat لكن زيادة العشوائية تساهم في زيادة تعميم الشبكة والقدرة على التصنيف على بيانات جديدة مما يساهم في رفع دقة الاختبار لاحقاً.

حددنا حجم بيانات التحقق (Validation) (أو الاختبار ضمن عملية التدريب) بنسبة 10% أي تقريبا 20238 عينة سيتم حساب دقة التدريب بالاختبار عليها، وسيتم التدريب على ما تبقى من عينات.

ويجدر بنا التنويه أن الفرق بين التحقق Validating والاختبار Testing أن التحقق هو اختبار أثناء عملية التدريب على شريحة من مجموعة بيانات التدريب يتم تحديدها، لإيجاد دقة التدريب بعد حساب الخطأ وإعادة ضبط أوزان وانحيازات الشبكة وتحقيق عملية التعلم، أما الاختبار Testing فيتم بعد عملية التدريب على بيانات مختلفة عن بيانات التدريب لإيجاد الدقة الفعلية للنموذج أو لإيجاد دقة الاختبار.

البارامتر الأخير الذي قمنا بتمريره callbacks=[checkpoint] يفيد في عملية حفظ النموذج عند نهاية كل epoch في حال تحسن الدقة.

model.save(config.model_path)

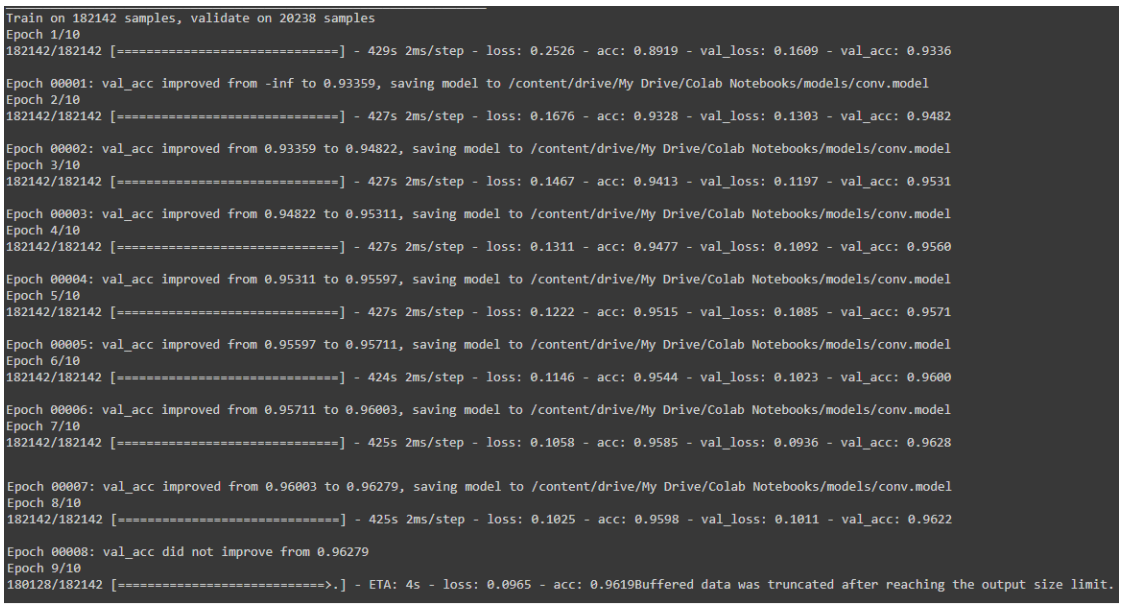
التعليمة السابقة تقوم بحفظ النموذج بشكل نهائي بعد الانتهاء من عملية التدريب.

عند التنفيذ سيظهر لدينا في الخرج بداية ملخص عن الشبكة بسبب تعليمة model.summary() الموجودة ضمن get_conv_model() فيها ملخص عن طبقات الشبكات وأبعاد خرج كل منها.

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 13, 9, 16)         160
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 13, 9, 32)         4640
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 13, 9, 64)         18496
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 13, 9, 128)        73856
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 6, 4, 128)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 6, 4, 128)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 3072)              0
_________________________________________________________________
dense_6 (Dense)              (None, 128)               393344
_________________________________________________________________
dense_7 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_8 (Dense)              (None, 2)                 130
=================================================================
Total params: 498,882
Trainable params: 498,882
Non-trainable params: 0
_________________________________________________________________
```

الشكل 5-11 ملخص عن الشبكة

ومن ثم ستبدأ عملية التدريب، استهلكت عملية التدريب لدينا حوالي الساعتين.



الشكل 5-12 خرج مرحلة التدريب

في السطر الأول نجد عدد العينات كما حسبناه سابقاً 202380 عينة كلية، 10% منها (20238 عينة) للاختبار، و90% منها (182142 عينة) للتدريب.

نلاحظ تزايد الدقة بعد كل دورة تدريب بداية 93.36% عند أول دورة تدريب وصولاً إلى 96.28% بعد دورة التدريب السابعة، بعد الدورة السابعة تثبت الدقة ولا تتحسن في الدورة الثامنة، أما في الدورة التاسعة فظهر لدينا الخطأ التالي:

Buffered data was truncated after reaching the output size limit.

بالبحث عن هذا الخطأ وجدنا أنه ناتج عن الحدود المفروضة من الذاكرة أثناء التدريب، أي بمعنى آخر تم استهلاك الذاكرة الممكنة بشكل كامل (12 GB مقدمة من خدمة Google Colab)، لكن نجد أيضاً أن الدقة الناتجة 96.28% ومن الممكن الاكتفاء بثماني دورات تدريب إذا حصلنا على نتائج جيدة، وبهذا نعلن انتهاء مرحلة التدريب.

5-7-اختبار الشبكة:

من أهم ما يجب دراسته عند اختبار نموذج تعلم عميق هو قدرة هذا النموذج على التعميم Generalization بحيث يقدم نتائج جيدة عند اختباره على بيانات جديدة، ويستطيع القيام بمهامه (تصنيف الجنس حسب الصوت في حالتنا) في ظروف مختلفة، فقد تحصل بعض النماذج على دقة تدريب مرتفعة ولكن عند اختبارها على بيانات جديدة مختلفة عن بيانات التدريب ينتج دقة اختبار سيئة، وعندها يكون النموذج يعاني من حالة Overfitting.

حصلنا على دقة تدريب 96.28%، سوف نختبر النموذج على بيانات جديدة ونوجد دقة الاختبار لكي نكون قادرين على تقييم قدرة النموذج على التعميم.

5-7-1-الاختبار على إشارات صوتية مسجلة بظروف مختلفة عن بيانات التدريب:

لكي نختبر النموذج نحتاج بيانات جديدة مختلفة عن بيانات التدريب ولذلك قمنا بتنزيل Dataset ثانية مختلفة عن التي استخدمناها في عملية التدريب (Pygender Audioset)، وهي تحوي 1104 تسجيل صوتي مستخرجة من مقاطع عشوائية منتشرة على الإنترنت، مسجلة بظروف عشوائية تحوي ضجيج وتم تسجيلها بميكروفونات مختلفة وبعضها داخل غرفة مغلقة والكثير منها في بيئة مفتوحة، وبعضها يحوي إشارات صوتية من أكثر من متكلم (هتافات، ضحك، أصوات جمهور) وقد لا تماثل ظروف تسجيل بيانات التدريب التي تم تسجيلها بميكروفون هاتف ذكي وداخل غرفة مغلقة، ولكن الاختبار على هذا النوع من البيانات والظروف القاسية سوف يعطينا فكرة عن أداء النموذج في تطبيقات واقعية مختلفة، علماً أننا سوف نختبر النموذج في مرحلة لاحقة على مجموعة بيانات اختبار أخرى قمنا بإنشائها نحن عبر تسجيل أصوات مجموعة من الأشخاص في ظروف مشابهة لظروف تسجيل بيانات التدريب.

عند تحميل الDataset من الإنترنت نجد لدينا مجلدين (female_clips – male_clips) كل منهما يحوي سجلات الصنف الخاصة به، ونلاحظ أن الملفات تمتلك أسماء عشوائية (مثال CHZ7OM1agU.wav).

في البداية نريد إعادة تسمية الملفات كل ملف حسب الصنف الذي ينتمي إليه وهي عملية غير ضرورية لكن تسهل علينا العمل، ومن ثم يجب علينا إنشاء ملف csv يحوي أسماء الملفات الصوتية والصنف المقابل لكل منها، يمكن إنجاز ذلك باستخدام المكتبة pandas كما يلي:

```
female_path="E:/GenderClassificationAudio/AudioSet/female_clips/"
male_path="E:/GenderClassificationAudio/AudioSet/male_clips/"
```

قمنا بتعريف مسار مجلدين كل منهما يحوي سجلات صنف من الأصناف.

```
for count, filename in enumerate(os.listdir(female_path)):
    if len(str(count)) == 1:
        base = "female_00"
    if len(str(count)) == 2:
        base = "female_0"
    if len(str(count)) == 3:
        base = "female_"
    dst = base + str(count) + ".wav"
    src = female_path + filename
    dst = female_path + dst
    os.rename(src, dst)
```

حلقة تمر على كافة ملفات التسجيل الأول وتقوم بإعادة تسمية كل منها على غرار النمط female_001 أو female_057 أو female_334.

```
for count, filename in enumerate(os.listdir(male_path)):
    if len(str(count)) == 1:
        base = "male_00"
    if len(str(count)) == 2:
        base = "male_0"
    if len(str(count)) == 3:
        base = "male_"
```

```
    dst = base + str(count) + ".wav"
    src = male_path + filename
    dst = male_path + dst
    os.rename(src, dst)
```

حلقة مشابهة للحلقة السابقة تقوم بإعادة تسمية ملفات الصنف الثاني.

```
female_list = os.listdir(female_path)
male_list = os.listdir(male_path)
```

قراءة محتويات المجلد الخاص بكل صنف وتخزين أسماء الملفات في list خاصة بكل صنف

```
fname = {'fname':female_list}
df1 = pd.DataFrame(fname)
df1['label'] = ['female']*len(df1)
```

تعريف Data Frame يحوي على أسماء ملفات الصنف الأول في العمود fname والقيمة female في العمود label.

```
fname = {'fname':male_list}
df2 = pd.DataFrame(fname)
df2['label'] = ['male']*len(df2)
```

تعريف Data Frame يحوي على أسماء ملفات الصنف الثاني في العمود fname والقيمة male في العمود label.

```
df = pd.concat([df1 , df2])
df.set_index('fname', inplace=True)
df.reset_index(inplace=True)
df.to_csv('test.csv' , index = False)
```

دمج كل من ال Data Frame الخاصة بسجلات كل صنف في Data Frame واحدة وحفظ النتيجة في ملف csv بالاسم test.

الآن أصبح لدينا مجموعة بيانات Data Set مشابهة للتي استخدمناها في عملية التدريب ويمكننا اختبار دقة الشبكة عليها، لتحقيق ذلك يجب علينا بدايةً أن نطبق عمليات Pre-processing التي طبقناها على بيانات التدريب وهي إزالة النقاط الميتة باستخدام التابع envelope وتخفيض تردد التقطيع إلى 16 KHz كذلك لأن استخدام تردد تقطيع مختلف سوف يؤثر على أبعاد خرج تابع MFCC (13×9) وتحديداً على قيمة عدد الإطارات (هنا 9) وبالتالي ستكون أبعاد البيانات لدينا مختلفة عن أبعاد البيانات التي دربنا عليها الشبكة وهذا لا يجوز ولن يتقبل النموذج حينها هذا الدخل، وبالتالي يجب التأكد من تطبيق نفس التردد المستخدم في عملية التدريب.

سوف نقوم بتعريف تابع نمرر له مسار يحوي على سجلات الاختبار ليقوم بإدخالها لنموذج الشبكة وتخزين النتائج كما يلي:

```
def build_predictions(audio_dir):
    y_true = []
```

تعريف List سوف نخزن فيها القيم الفعلية للأصناف

```
    y_pred = []
```

تعريف List سوف نخزن فيها القيم المتوقعة للأصناف

```
fn_prob = {}
```
تعريف قاموس Dictionary سنخزن فيه أسماء سجلات الاختبار والقيم المتوقعة المقابلة لها
```
print('Extracting features from audio')
for fn in tqdm(os.listdir(audio_dir)):
```
حلقة تمر على كل عناصر المسار الذي يحوي على بيانات الاختبار
```
    signal , rate = librosa.load(os.path.join(audio_dir, fn), sr=16000)
    mask = envelope(signal, rate, 0.0005)
    wav = signal[mask]
```
قراءة الإشارة الصوتية وفق التردد 16 KHz وإزالة النقاط الميتة منها وتخزين الإشارة بعد التنظيف في wav
```
    label = fn2class[fn]
    c = classes.index(label)
```
قراءة قيمة الصنف المقابلة للإشارة المقروءة سابقاً (القيمة الفعلية للخرج)
```
    y_prob = []
```
تعريف List سوف نخزن فيها القيمة المتوقعة لاحتمال انتماء الدخل لكل من الصنفين
```
    for i in range(0, wav.shape[0]-config.step, config.step):
```
حلقة تمر على السجل الصوتي المقروء بحيث ننتقل ضمنه إطار تلو الإطار
```
        sample = wav[i:i+config.step]
```
عند كل إطار سوف نأخذ عينة
```
        x = mfcc(sample, rate, numcep=config.nfeat,
                 nfilt=config.nfilt, nfft=config.nfft).T
```
تمرير العينة لتابع MFCC لاستخراج السمات من العينة الواردة
```
        x = (x - config.min)/(config.max - config.min)
```
تقييس الدخل قبل إدخاله للشبكة (قيمة السمات هنا X هي الدخل)
```
        x = x.reshape(1, x.shape[0], x.shape[1], 1)
```
إعادة تشكيل أبعاد X بحيث تتوافق مع الأبعاد الذي دربنا عليها الشبكة في مرحلة التدريب، يجدر بنا الذكر هنا أن الواحد الذي على اليمين هو نفسه الواحد الذي قمنا بإضافته في مرحلة التدريب وقلنا حينها أنه ضروري وفق بنية شبكات CNN، أما الواحد الذي على اليسار فهو مقابل لعدد عينات الدخل في مرحلة التدريب حيث كانت أبعاد الدخل حينها بالضبط (1×9×13×182142) وكان حينها العدد 182142 يعبر عن عدد عينات التدريب، أما هنا نقوم بإدخال عينة واحدة لذلك نضيف بعد إضافي قيمته واحد.
```
        y_hat = model.predict(x)
```
إيجاد خرج النموذج وهنا يجب التنويه كما أشرنا سابقاً أن الخرج سيكون بالشكل Hot Encoding أي عبارة عن مركبتين كل مركبة تعبر عن احتمال انتماء هذه العينة لصنف من الأصناف.
```
        y_prob.append(y_hat)
```
تخزين احتماليتي الانتماء لكل صنف من الأصناف ضمن اللائحة y_prob
```
        y_pred.append(np.argmax(y_hat))
```
إجراء عكس لترميز Hot Encoding باستخدام تابع argmax كمان قمنا في عملية التدريب للحصول على الاحتمال الأكبر وإيجاد تصنيف الشبكة النهائي لهذه العينة وتخزين الناتج ضمن اللائحة y_pred
```
        y_true.append(c)
```

تخزين القيمة الفعلية لهذه العينة ضمن اللائحة y_true لكي نتمكن من مقارنتها مع القيمة المتوقعة لاحقاً.

وتكرار حلقة for الداخلية الخاصة بالمرور على عينات كل سجل.

```
fn_prob[fn] = np.mean(y_prob, axis=0).flatten()
```

بعد المرور على كل السجلات سنقوم بتخزين النتائج ضمن ال Dictionary (fn_prob)

```
return y_true, y_pred, fn_prob
```

انهاء التابع وإعادة النتائج.

الآن اعتماداً على التابع السابق سوف نقرأ الملفات الصوتية ونوجد تصنيف الشبكة لها ونحسب دقة الاختبار كما يلي:

```
df = pd.read_csv('test.csv')
```

قراءة ملف csv الذي يحوي على أسماء ملفات الاختبار والأصناف الموافقة لكل منها.

```
classes = list(np.unique(df.label))
```

تعريف list بالأصناف (Female-Male)

```
fn2class = dict(zip(df.fname, df.label))
```

تعريف قاموس Dictionary يحوي اسم كل سجل والصنف المقابل له

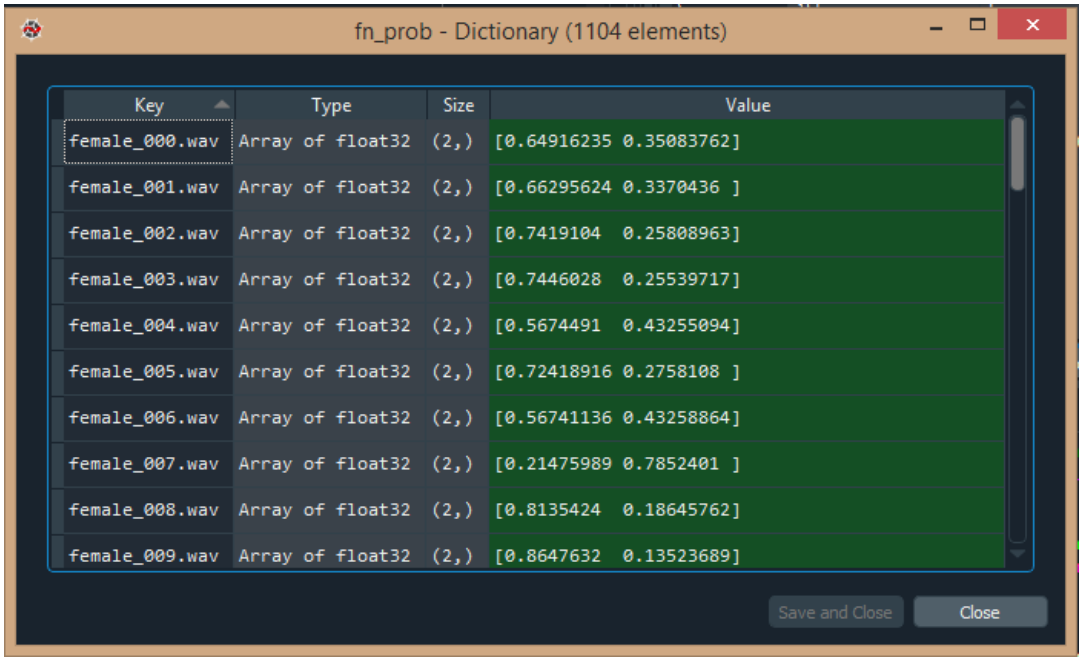
```
p_path = os.path.join('pickles', config.mode+'.p')
with open(p_path, 'rb') as handle:
    config = pickle.load(handle)
model = load_model('models/'+ config.mode +'.model')
```

استدعاء النموذج الذي دربناه سابقاً

```
y_true, y_pred, fn_prob = build_predictions('test')
```

استدعاء التابع الذي يقوم بإدخال العينات للشبكة وتخزين النتائج.

الآن لدينا Dictionary يحوي أسماء السجلات ومصفوفة ثنائية مقابلة لكل سجل تحوي احتمال انتماء هذا السجل لكل من الصنفين كما في الشكل (5-13):



الشكل 5-13 سجلات الاختبار واحتماليات انتمائها لكل صنف

```
y_probs = []
```

نقوم بتعريف لائحة List ونمر على جميع السجلات بحلقة for

```
for i, row in df.iterrows():
   y_prob = fn_prob[row.fname]
   y_probs.append(y_prob)
```

تخزين احتمال انتماء كل سجل للأصناف في اللائحة التي عرفناها سابقاً

```
   for c, p in zip(classes, y_prob):
      df.at[i, c] = p
```

إضافة أعمدة جديدة إلى Data Frame التي قرأناها من ملف csv سابقاً، تحوي هذه الأعمدة على احتماليات انتماء السجل لكل صنف من الأصناف.

```
y_pred = [classes[np.argmax(y)] for y in y_probs]
```

إيجاد القيمة الأكبر بين الاحتمالين عند كل سجل وتخزين التصنيف النهائي في لائحة List

```
df['y_pred'] = y_pred
```

إنشاء عمود جديد ضمن ال Data Frame التي قرأناها سابقاً من ملف csv تحوي على التصنيف النهائي لكل سجل اعتماداً على الاحتمال الأكبر.

```
acc_score = accuracy_score(y_true=df['label'], y_pred=df['y_pred'])
print("\n accuracy :" , acc_score*100, "%")
```

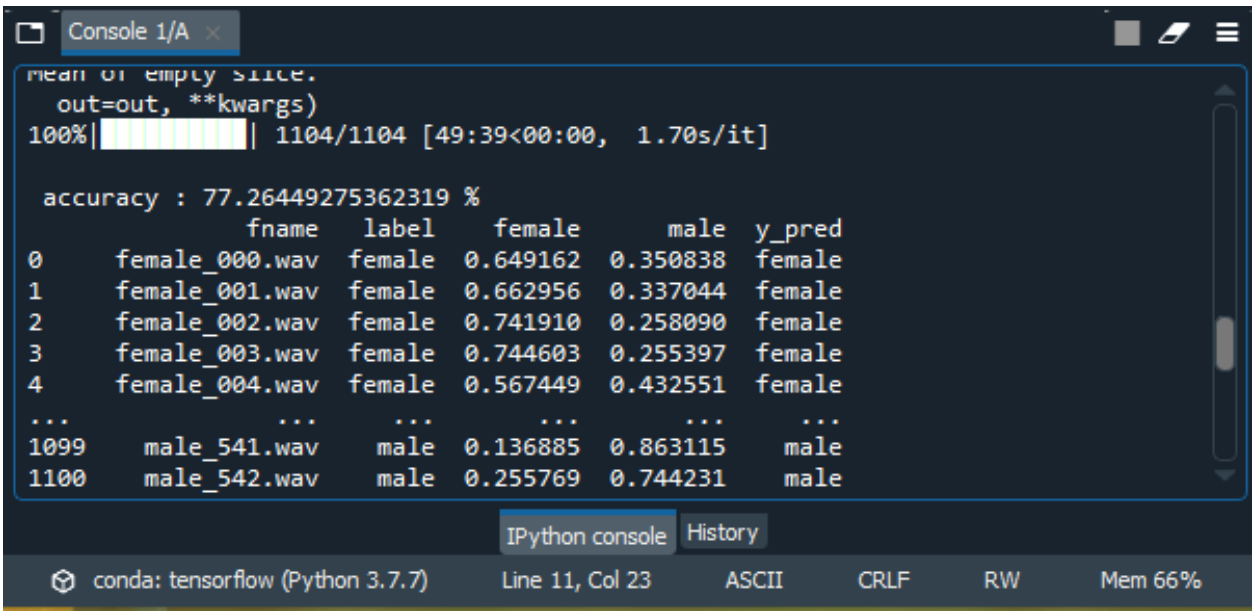
حساب الدقة وفق معيار Accuracy (شرحناه سابقاً في مرحلة التدريب) وطباعة النتيجة

```
df.to_csv('predictions.csv', index=False)
print(df)
```

تخزين النتائج في ملف csv منفصل.

عند تنفيذ البرنامج السابق حصلنا على دقة اختبار 77.26%، وملف predicions.csv يحتوي على النتائج.

كما نلاحظ أثناء التنفيذ أن النموذج استهلك بين 1 إلى 2 ثانية لتصنيف كل سجل علماً أن أغلب الملفات الصوتية طول كل منها حوالي 10 ثواني، كما أننا نقوم أثناء التنفيذ بتنظيف البيانات (تخفيض دقة التقطيع وإزالة النقاط الميتة) بالإضافة لتمرير ناتج عملية التنظيف للشبكة لمعالجتها، نتوقع من النتائج الحالية إمكانية تطبيق نموذج الشبكة ضمن منظومة تعرف على جنس المتكلم بالزمن الحقيقي أو على الأقل نتوقع أداء جيد للنموذج ضمن تطبيقات تتطلب سرعة تصنيف (Live Gender Audio Classification)، مع بعض التحسينات والإضافات في المستقبل.



الشكل 5-14 دقة الاختبار الناتجة عن مجموعة البيانات الأولى

E	D	C	B	A	
y_pred	male	female	label	fname	1
female	0.350838	0.649162	female	female_000.wav	2
female	0.337044	0.662956	female	female_001.wav	3
female	0.25809	0.74191	female	female_002.wav	4
female	0.255397	0.744603	female	female_003.wav	5
female	0.432551	0.567449	female	female_004.wav	6
female	0.275811	0.724189	female	female_005.wav	7
female	0.432589	0.567411	female	female_006.wav	8
male	0.78524	0.21476	female	female_007.wav	9
female	0.186458	0.813542	female	female_008.wav	10
female	0.135237	0.864763	female	female_009.wav	11
male	0.622003	0.377997	female	female_010.wav	12
female	0.08518	0.91482	female	female_011.wav	13
female	0.338704	0.661296	female	female_012.wav	14
female	0.01809	0.98191	female	female_013.wav	15
female	0.112406	0.887594	female	female_014.wav	16
male	0.621477	0.378523	female	female_015.wav	17
male	0.621784	0.378216	female	female_016.wav	18
female	0.461539	0.538461	female	female_017.wav	19
female	0.224949	0.775051	female	female_018.wav	20
female	0.316925	0.683075	female	female_019.wav	21
male	0.589677	0.410323	female	female_020.wav	22
female	0.337528	0.662473	female	female_021.wav	23
female	0.206029	0.793971	female	female_022.wav	24
female	0.495226	0.504774	female	female_023.wav	25

predictions

الشكل 5-15 خرج مرحلة الاختبار على مجموعة البيانات الأولى

5-7-2-مناقشة النتائج:

بالنظر لجدول النتائج في نهاية الفصل ونسبة الأخطاء في كل صنف، نلاحظ أن نسب التصنيفات الخاطئة متقاربة بين الصنفين، لذلك يمكن أن نستبعد أن تكون الأخطاء ناتجة عن تعلم النموذج لسمات صنف معين دون الآخر.

عند مراجعة بعض السجلات التي أخطأ النموذج في تصنيفها وجدنا أن الكثير منها يحوي أصوات هتافات وتداخل بين أكثر من متكلم مثل التسجيلات: female_322 , female_333, female_345, female_405, male_059, male_067, male_454, male_384، وبعض التسجيلات تحوي أصوات ضحك فقط دون تكلم مثل: male_453، female_310، كما وجدنا تسجيل فارغ لا يحوي أي صوت male_424، ووجدنا تسجيل يحوي صوت لما يبدو لشخص يحاول تقليد صوت بطة male_493، حيث أن ال Data Set الخاصة بالاختبار كما ذكرنا مأخوذة من تسجيلات عشوائية لمقاطع منشورة على شبكة الإنترنت.

هنا نجد أن الكثير من الأخطاء التي وقع فيها النموذج ناتجة عن سوء تسجيل بيانات الاختبار، ونسبة الضجيج المرتفعة فيها، فبالعودة لبيانات التدريب، كنا قد دربنا النموذج على Data Set لتسجيلات صوتية مسجلة ضمن غرفة مغلقة وبيئة هادئة، وبميكرفون هاتف ذكي ذو جودة مقبولة.

نتوقع أن يقدم نموذجنا نتائج أفضل عند اختباره على بيانات بشروط تسجيل مقبولة وأكثر جودة مشابهة لشروط تسجيل بيانات الاختبار، لكن بالمقابل أيضاً، اختبار النموذج على البيانات السابقة يسمح لنا بتقييم قدرته على التعميم، وتوقع سلوكه في ظروف تسجيل رديئة أحياناً، لأخذ صورة عن أدائه في ظروف قاسية وأكثر واقعية.

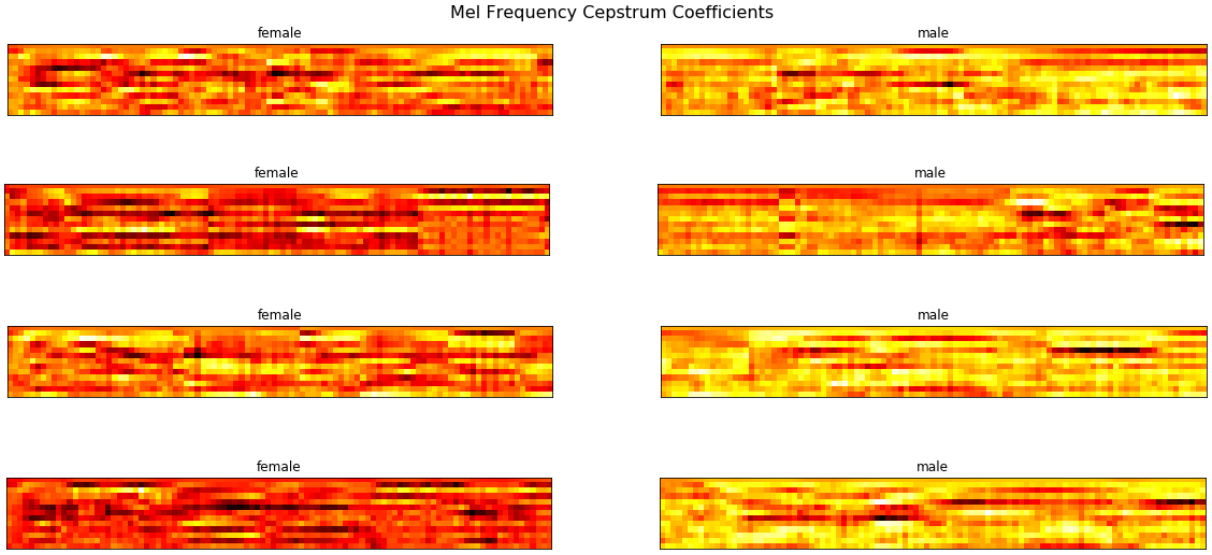
5-7-3-الاختبار على إشارات صوتية مسجلة بظروف مشابهة لبيانات التدريب:

بعد اختبار النموذج على مجموعة بيانات مسجلة بظروف مختلفة عن ظروف تسجيل بيانات التدريب، يشوبها الضجيج بنسبة مرتفعة، قررنا اختبار النموذج على بيانات جديدة لكن مشابهة بظروف التسجيل لظروف تسجيل بيانات التدريب، لتقييم أداء النموذج عند دخل يحوي نسبة ضجيج مشابهة لنسبة ضجيج تسجيلات بيانات التدريب، لذلك طلبنا من عدة متطوعين (مجموعة من زملائنا الطلاب في جامعة تشرين) تسجيل جمل وعبارات مختلفة بأصواتهم، في غرفة مغلقة وبيئة هادئة باستخدام ميكروفونات هواتفهم الذكية، وقمنا بإنشاء مجموعة بيانات صغيرة خاصة بنا تحوي 38 سجل، 20 سجل تنتمي للصنف Female، و18 سجل تنتمي للصنف Male، تتميز بترددات تقطيع متنوعة (16 KHz - 44.1 KHz)، وبأطوال ملفات متنوعة (من 4 ثواني إلى 20 ثانية)، مسجلة بميكروفونات هواتف مختلفة وفي غرف وأماكن مختلفة بشرط أن تكون الغرفة مغلقة والبيئة المحيطة هادئة نسبياً، وقمنا بتحويل جميع الملفات الصوتية للصيغة wav، لاختبارها على النموذج وتقييم الأداء في ظروف تسجيل مقبولة.

5-7-4-جدول النتائج النهائية:

مجموعة البيانات	Free ST American Corpus (90% تدريب و10% تحقق)	Pygender Audio Set (100% اختبار)	مجموعة البيانات الخاصة بنا (100% اختبار)
عدد السجلات الكلي	3843 سجل	1104 سجل	40 سجل
عدد سجلات الصنف Female	2186 سجل (%56.88)	558 سجل (%50.54)	20 سجل (%50)
عدد سجلات الصنف Male	1656 (%43.12)	546 سجل (%49.46)	20 سجل (%50)
نسبة السجلات التي تم تصنيفها بشكل صحيح	%96.28 (دقة التحقق)	853 سجل %77.26	%100
نسبة السجلات التي تم تصنيفها بشكل خاطئ	%3.72 (خطأ تحقق)	251 سجل %22.78	%0
نسبة سجلات الصنف Female التي تم تصنيفها بشكل صحيح		419 سجل (%75.08)	%100
نسبة سجلات الصنف Male التي تم تصنيفها بشكل صحيح		434 سجل (%79.48)	%100
نسبة سجلات الصنف Female التي تم تصنيفها بشكل خاطئ		139 سجل (%24.91)	%0
نسبة سجلات الصنف Male التي تم تصنيفها بشكل خاطئ		112 سجل (%20.51)	%0

يجدر بنا الذكر هنا أننا حاولنا تحقيق النظام باستخدام نموذج شبكة عصبونية تكرارية قبل استخدامنا لنموذج شبكة عصبونية التفافية (يمكن الاطلاع على النص البرمجي الخاص بنموذج الشبكة التكرارية في الملحق الموجود في نهاية هذا التوثيق)، تستخدم الشبكات التكرارية في العديد من تطبيقات معالجة الصوت والتعرف على اللغات الطبيعية، حيث تسمح بنيتها بمعالجة السلاسل الزمنية والتنبؤ بقيمها، لم يحقق نموذج الشبكة التكرارية دقة جيدة مقارنة بنموذج الشبكة الالتفافية، وجدنا أن السبب الرئيسي لنجاح نموذج الشبكة العصبونية الالتفافية أننا تمكنا باستخدام خوارزمية MFCC من تحويل الإشارات الصوتية إلى أنماط يمكن تمييز الكثير منها بالعين المجردة حتى وتوقع انتمائها لأحد الأصناف المدروسة، انظر الشكل (5-16)، وبالتالي حولنا المسألة المدروسة لمسألة شبيهة بدراسة أنماط الصور والمخططات الرسومية والتي تمتاز الشبكات العصبونية الالتفافية بقدرتها على تعلم سماتها.



الشكل 5-16 خرج MFCC لعدة سجلات من مجموعة بيانات التدريب

6-الفصل السادس: المشاكل التي واجهتنا والتطوير المستقبلي

6-1-المشاكل التي واجهتنا:

- عدم التوازن بين الأصناف (Class Imbalance) ضمن مجموعة بيانات التدريب:

نتجت هذه المشكلة عن أن نسبة كمية البيانات ضمن مجموعة بيانات التدريب للصنف Male أكبر من نسبة كمية بيانات التدريب الصنف Female، وبالتالي قد يتعلم النموذج لدينا على سمات صنف محدد بنسبة أكبر من الصنف الآخر، مما يؤثر سلباً على أداء النموذج.

وقمنا بحل هذه المشكلة بمنهجية انتقاء عيناء عشوائية للتدريب وفق نسبة توزع البيانات بين الصنفين، كما ساعدنا تابع compute_class_weight من مكتبة scikit-learn على حساب الأوزان أثناء عملية التعلم بشكل يحقق توازن في التدريب والتعلم على بيانات كل صنف.

- صعوبة التدريب والتنفيذ على الأجهزة:

تستهلك عملية تدريب الشبكة العصبونية الكثير من الوقت والموارد وقد ترهق العتاد الصلب للحاسب، وقد يستهلك تنظيف البيانات واختبار النموذج كذلك وقتاً وموارد، لذلك قمنا بتنظيف البيانات والتدريب على GPU مقدم من خدمة Google Colab، وحاولنا استغلال ما يمكن توفيره من إمكانيات.

- صعوبة الحصول على بيانات عالية الجودة:

لم يتوفر لدينا بيانات اختبار أفضل من التي اختبرنا عليها فأغلب مجموعات البيانات Data Sets المتوفرة على الانترنت ضخمة جدا (بأحجام 10 او 20 GB) وتستهلك وقتاً وموارد تدريب أكثر مما قد يمكن أن توفره لنا خدمة Google Colab، نتوقع أن يقدم نموذجنا نتائج أفضل عند اختباره على بيانات بشروط تسجيل مقبولة وأكثر جودة مشابهة لشروط تسجيل بيانات الاختبار، لكن بالمقابل أيضاً، اختبار النموذج على البيانات السابقة يسمح لنا بتقييم قدرته على التعميم، وتوقع سلوكه في ظروف تسجيل رديئة أحياناً، لأخذ صورة عن أدائه في ظروف قاسية وأكثر واقعية.

6-2-التطوير المستقبلي:

- تصنيف حي Live Classification ضمن الزمن الحقيقي (Real-time):

وجدنا عند الاختبار أن النموذج يحتاج بضع ثواني (بين ثانية واحدة وثانيتين) لتصنيف تسجيل صوتي بطول 10 ثواني، ربما يسمح هذا في المستقبل بتطبيق النموذج ضمن نظام تصنيف حي أو بالزمن الحقيقي إن أمكن، مع أضافة بعض التحسينات لاختصار الزمن اللازم لتنظيف البيانات قبل إدخالها للشبكة، لتخفيض الوقت الكلي اللازم لعملية المعالجة.

- التدريب على بيانات مسجلة بظروف مختلفة:

عند اختبار النموذج على مجموعة الاختبار الأولى (Pygender Audio Set)، وجدنا أن النتائج تأثرت بظروف التسجيل، نقترح تدريب النموذج على بيانات مختلفة بظروف مختلفة وميكروفونات مختلفة وأكثر تنوعاً، لتعليم النموذج على التصنيف ضمن بيئة تسجيل تحوي ضجيج أكثر، لتحسين الأداء ضمن الظروف الواقعية والإشارات الصوتية العشوائية، يعد تحليل وتنظيم البيانات علماً مستقلاً بحد ذاته وقد يكون خارج صلب دراستنا، على العموم، نقترح إمكانية تنظيف سجلات مجموعة البيانات (Pygender Audio Set) التي اختبرنا عليها من الضجيج والسجلات الرديئة، واستغلالها لتدريب النموذج.

- ربط النموذج بواجهة رسومية:

يمكن ربط النموذج بواجهة مستخدم رسومية لتسهيل التعامل في التطبيقات الواقعية، وتحوي لغة Python العديد من المكاتب المستخدمة لبناء واجهات المستخدم GUI، مثل Tkinter، Kivy، PyQT، WxPython، Libavg، PySide، PyGUI، Pyforms، Wax Python GUI.

ملحق (نموذج الشبكة العصبونية التكرارية المستخدم)

```
def get_recurrent_model:()
#    shape of data for RNN is (n , time, feat)
    model = Sequential()
    model.add(LSTM(128, return_sequences=True, input_shape=input_shape))
    model.add(LSTM(128, return_sequences=True))
    model.add(Dropout(0,5))
    model.add(TimeDistributed(Dense(64, activation='relu')))
    model.add(TimeDistributed(Dense(32, activation='relu')))
    model.add(TimeDistributed(Dense(16, activation='relu')))
    model.add(TimeDistributed(Dense(8, activation='relu')))
    model.add(Flatten())
    model.add(Dense(2, activation='softmax'))
    model.summary()
    model.compile(loss='categorical_crossentropy,'
                  optimizer='adam,'
                  metrics=['acc'](
  return model
```

المراجع

[1] د. علي، الشبكات العصبية الصناعية خوارزميات التعليم – تطبيق التعليم بدون معلم على الشبكات العصبية الصناعية، 2018.

[2] L. Rabiner و R. Schafer، Introduction to digital speech processing، 2007.

[3] "https://towardsdatascience.com/conv-nets-for-dummies-a-bottom-up-approach-c1b754fb14d6،" [متصل]. [تاريخ الوصول 2020].

[4] "https://medium.com/@udemeudofia01/basic-overview-of-convolutional-neural-network-cnn-4fcc7dbb4f17،" [متصل]. [تاريخ الوصول 2020].

[5] "https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html،" [متصل]. [تاريخ الوصول 2020].

[6] "http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/،" [متصل]. [تاريخ الوصول 2020].

[7] "http://datagenetics.com/blog/november32012/index.html،" [متصل]. [تاريخ الوصول 2020].

[8] م. م. م. المالح، تلخيص النصوص العربية المعتمد على التعلم العميق والمعطيات الكبيرة، دمشق، الجمهورية العربية السورية: المعهد العالي للعلوم التطبيقية والتكنولوجيا، 2019.

Tishreen University
Faculty of Mechanical and Electrical Engineering
Department of Computer and Automatic Control Engineering
Fifth Year

Gender Classification Based on Voice Using Convolutional Neural Networks (CNN)

Prepared to obtain License in Computer and Automatic Control Engineering

Prepared by:

Ahmad Mahmoud Shahla **Anwar Firas Makhous**

Aous Mohammed Hasan

Supervised by:

Dr. Majd Ali

2019-2020