ChatGPT

ASVspoof Challenge and Anti-Spoofing in Speaker Verification

Introduction to ASV Spoofing Detection

Automatic Speaker Verification (ASV) systems aim to verify a speaker's identity from their voice, but they are vulnerable to **spoofing attacks** where an impostor's audio is presented as someone else's voice [1] [2]. **ASVspoof detection** refers to the techniques and countermeasures used to distinguish between **bona fide** (genuine) speech and **spoofed** or AI-generated speech. Spoofing can take several forms: **text-to-speech (TTS) synthesis**, where entirely artificial speech is generated; **voice conversion (VC)**, where one person's voice is transformed to sound like another; and **replay attacks**, where a recording of a genuine speaker is played back to the system [3] [4]. In all cases, the attacker's goal is to deceive the ASV system into falsely accepting the spoofed audio as a genuine user.

Detection of such spoofing attacks is essentially a specialized audio classification problem. The system (often called a *spoofing countermeasure* or *presentation attack detector*) takes an input audio waveform and produces a score or decision indicating whether the speech is real or fake [1] [2]. Early approaches relied on handcrafted signal features and simple classifiers, but modern systems leverage deep learning to automatically learn discriminative features that highlight artifacts of spoofing. Spoofing detection typically operates as a **front-end to an ASV system**: the countermeasure screens input audio, and only if it is classified as bona fide will it be passed to the ASV for identity verification [5] [4]. This tandem operation ensures that even if an attacker can perfectly mimic a target speaker's voice, the synthetic or replay nature of the audio can be caught by the countermeasure.

The ASVspoof challenge series (started in 2015) was established to benchmark progress in this field [6] [7]. By providing common datasets of genuine and spoofed speech and a standardized evaluation, ASVspoof challenges allow researchers to compare solutions and track improvements over time [8] [9]. In the following sections, we provide a deep dive into the 2019 and 2021 ASVspoof challenge editions – their tasks, datasets, evaluation protocols, and outcomes – and discuss how anti-spoofing techniques evolved between these editions. We also outline state-of-the-art methods for spoof detection and give practical guidance for building a system to detect AI-generated (spoofed) audio.

ASVspoof 2019 Challenge Overview

Tasks and Dataset in 2019

The **ASVspoof 2019** challenge introduced two core sub-challenges, each addressing a different attack scenario [10] [11]:

- **Logical Access (LA):** A scenario focused on *remote* attacks, where an attacker injects spoofed audio directly into the communication channel of a voice authentication system (bypassing any microphone) [12] [3]. This is analogous to telephone-based or server-based authentication being

attacked with synthesized or converted speech. The LA dataset consisted of speech from the **VCTK corpus** (multi-speaker English database) that had been manipulated by a diverse set of **17 TTS and VC algorithms** [13] [3]. These ranged from waveform concatenation methods to advanced neural synthesizers like WaveNet [14]. To simulate real conditions, the attacks were divided into *6 known* attack types and *11 unknown* attack types [3]. Systems A1–A6 (2 VC and 4 TTS) were the **known attacks** used to generate training and development data, while systems A7–A19 (including additional TTS, VC, and hybrid TTS-VC methods) were **unknown attacks** that appeared only in the evaluation set [3] [15]. This meant participants trained countermeasures on a limited set of attacks and were tested on their ability to generalize to new spoofing methods. All LA audio was **clean speech** (recorded in a hemi-anechoic chamber without added noise or channel effects).

- **Physical Access (PA):** A scenario addressing *physical* or *presentation* attacks, specifically replay attacks [16] [17]. Here, an attacker covertly records a target speaker's voice and then plays it back to the ASV system via a speaker in a physical environment (for instance, trying to unlock a smartphone's voice authentication by playing a recording). The PA dataset was a **carefully simulated** collection of replay attacks that model various acoustic conditions and attacker capabilities [16] [18]. The simulation followed the ISO standard definition of presentation attacks, varying factors such as: (a) the **room size (S)** – small, medium, large rooms (from ~2–5 m^2 up to 10–20 m^2) [19]; (b) the **reverberation time (T60)** – low, mid, high (ranging from ~50–200 ms up to 600–1000 ms) [20]; and (c) the **distances** – the attacker's recording distance from the talker (D_a) and the playback distance to the ASV microphone (D_s), each categorized as low (10–50 cm), medium (50–100 cm), or large (100–150 cm) [20] [17]. In addition, the **playback device quality (Q)** was varied: perfect (an ideal flat-response device), high-quality, or low-quality playback equipment [18]. The combination of these factors produced **27 possible acoustic environments** (room size × reverberation × mic distance) and **9 replay attack configurations** (attacker distance × device quality), labeled with environment identifiers (e.g., "aaa" for a particular combo) and attack identifiers ("AA" through "CC") [20] [18]. For example, one attack might involve a high-quality speaker replaying audio in a large room from 1.5 m away, while another might use a low-quality phone speaker in a small room from 0.3 m away. By simulating these systematically, the PA database captured a wide variety of replay conditions that an ASV system might encounter [20] [18]. Importantly, all PA data was **simulated using software** (Roomsimove and a polynomial Horn loudspeaker model) rather than actual re-recordings [21] [20]. This allowed tight control over parameters and ensured reproducibility. *(Note: After the 2019 challenge, a small real PA dataset was later released for analysis, but the official 2019 evaluation used only simulated replays.)*

Each sub-challenge had separate training, development, and evaluation partitions. The LA data was derived from 107 speakers (46 male, 61 female) in VCTK, with specific subsets for train/dev/eval (e.g. 20 train speakers, 10 dev, rest eval, ensuring disjoint sets) [22]. The PA data similarly had a partitioning where no speaker's voice (and no specific replay configuration) was duplicated across training and evaluation sets, to avoid overly optimistic results [23] [24]. In total, the 2019 database was significantly larger and more diverse than previous editions (ASVspoof 2015/2017) – containing on the order of tens of thousands of trials per condition – which enabled training data-hungry models like neural networks [11] [25].

Evaluation Protocol and Metrics in 2019

To evaluate spoofing countermeasures, ASVspoof 2019 introduced a new primary metric, the **tandem detection cost function (t-DCF)** [26] [27]. The t-DCF is an ASV-centric metric that considers the effect of the

spoof detector on the overall verification system's security. It essentially combines the *spoof miss rate* (letting a fake through) and the *spoof false alarm rate* (incorrectly blocking a genuine user) with the underlying ASV system's own false accept/reject rates [26]. This provides a single normalized cost value: a *min t-DCF of 0* would indicate a perfect countermeasure (no added cost to the ASV system from spoofing), while *1.0* would indicate a countermeasure no better than doing nothing (i.e., spoofing attacks would fully compromise the ASV at target performance) [26]. In simpler terms, t-DCF measures how much the spoofing detector improves security when it is "tandem'ed" with an ASV system. The ASVspoof organizers fixed a reference ASV system (a state-of-art speaker embedding/PLDA system) and provided its scores, so all participants' t-DCF calculations used a common ASV baseline [27] [28]. In addition to t-DCF, the standard **Equal Error Rate (EER)** was reported as a secondary metric for convenience and continuity with past challenges [26]. EER is the rate at which the false acceptance and false rejection rates are equal, treating spoof detection as a binary classification.

Participants in ASVspoof 2019 had to submit system output scores on both the development set (for validation) and the held-out evaluation set, for each task (LA and PA). They were allowed up to **four submissions per task**, which typically included: one *primary system* (often an ensemble of subsystems, allowed to use fusion of multiple models), one *single system* (a single-model result from among the ensemble, used for ranking of single systems), and optionally two contrastive systems with alternative approaches [29] [30]. The **rules** explicitly forbade the use of **external speech data** for training – systems had to be trained only on the provided ASVspoof 2019 train (and optionally dev) data [31] [30]. However, an exception was made allowing participants to use external *non-speech* resources for data augmentation [32] [33]. This meant one could add noise, impulse responses, or apply simulation tools (like codec compression, room simulation) to the existing data to better match evaluation conditions, as long as no additional labeled speech corpora were used. This rule anticipated that generalization to unseen attacks might be improved by augmenting the training with varied conditions (a principle that would become even more important in 2021). All systems were evaluated uniformly by the organizers, and final rankings were primarily based on **minimum normalized t-DCF** (the minimum t-DCF achieved after choosing an optimal score threshold) on the evaluation set [34] [35].

Baseline Systems in 2019

The challenge provided two baseline countermeasure (CM) systems to serve as reference points [36] [37]:

- **Baseline B01: CQCC-GMM.** This system used **Constant-Q Cepstral Coefficients (CQCC)** features with a Gaussian Mixture Model backend [36] [37]. CQCCs are a spectral feature extracted using a constant-Q transform (a wavelet-like time-frequency analysis with a logarithmic frequency scale). The baseline configuration extracted CQCC features from 15 Hz–8 kHz with 96 bins per octave, yielding 29 cepstral coefficients (plus the 0th) which were then augmented with delta and delta-delta, resulting in a 90-dimensional feature vector per frame [36] [38]. The classifier was a simple **GMM** (512 mixture components) trained separately on bona fide and spoof examples using expectation-maximization [37]. At test time, an utterance's score was the log-likelihood ratio between the bona fide-model and spoof-model likelihoods [37].

- **Baseline B02: LFCC-GMM.** This system used **Linear Frequency Cepstral Coefficients (LFCC)** features, also with a GMM backend [39] [37]. LFCCs are essentially conventional cepstral coefficients computed on a linear frequency scale (as opposed to the mel scale in MFCCs). The baseline extracted 20 ms frames (50% overlap) and computed 19 cepstral coefficients (plus 0th) from a 512-point FFT

covering 30 Hz–8 kHz [39]. Including delta and delta-delta, this made a 60-dimensional feature per frame [40]. The classifier setup (512-component GMMs for real vs spoof) was the same as B01 [37].

These baseline systems were essentially **carry-overs from prior work** – CQCC and LFCC with GMM had been top performers in earlier challenges (2015/2017). They provided a benchmark for participants to beat. For context, on the 2019 evaluation data, the baselines achieved around **8–10% EER** in the LA scenario and higher in PA. Indeed, the LA development EERs for B01 and B02 were reported as 9.57% and 8.09% respectively [35], and the PA baseline EERs were even higher (reflecting the challenging variability in replay data). These relatively high error rates indicated that there was ample room for improvement with more advanced techniques.

(Side note: The organizers released an open-source MATLAB/Python toolkit for computing t-DCF and the baselines, enabling participants to validate their implementations [27] [41]. This helped ensure everyone calculated t-DCF correctly and could reproduce baseline results.)

Results and Winning Systems in 2019

ASVspoof 2019 saw a large number of submissions (48 for LA, 27 for PA, from 23 research teams) and delivered substantial improvements over the baselines [42] [43]. The top systems achieved remarkably low error rates, especially in the logical access scenario:

- **Logical Access (LA) Results:** The best primary system for LA (denoted team *T05*) obtained a **minimum t-DCF of 0.0692** and an **EER of only 0.22%** on the evaluation set [44]. This EER – essentially zero – means it correctly identified all but 0.22% of trials, a huge leap from the ~8–9% EER of the baseline [35] [44]. In fact, the T05 system's t-DCF was *only marginally above the "ASV floor"* of 0.0627 (the theoretical minimum cost if a perfect detector were used) [44] [45]. This indicates that in the LA scenario (clean, electronic injection attacks), current countermeasures can almost completely mitigate spoofing with negligible impact on genuine trials. Other top single systems (e.g., T45, T24) had EERs around 1.9–5% [35], still a solid improvement over baselines. Notably, even the best *single* system (T45, with EER 1.86%) was significantly outperformed by the *ensemble* primary (T05) due to fusion gains [42] [46].

Top System Techniques (LA): A common theme among top LA systems was the use of **advanced front-ends and deep neural networks**, often fused in ensembles. For example, the winning T05 primary system was a **fusion of seven subsystems** [43] [47]. Six subsystems used different **spectral representations** derived from the Fourier transform (with various window lengths: 256, 160, 128 samples) to capture artifacts at multiple time-frequency resolutions [43]. The seventh subsystem used a **discrete cosine transform (DCT)** on those spectra (essentially a cepstral feature, possibly an inverted MFCC focusing on high-frequency content) [43]. These features were fed into an array of classifiers: the T05 fusion included **MobileNet, DenseNet, and ResNet-50 CNNs** as well as potentially an SVM or GMM for the DCT-based features [43] [48]. The outputs of the subsystems were combined by **score averaging or logistic regression** to produce the final decision [49] [50]. Another top system (T45 single) used a single-model approach with a **Light CNN (LCNN)** operating on LFCC features and an angular margin softmax loss, which already halved the EER of the LFCC-GMM baseline [51] [45]. Many teams incorporated novel input features like **CQCC, inverted MFCC (IMFCC), group delay grams, or constant-Q transform (CQT) spectrograms**, combined with various CNN architectures [52] [53]. Four of the top five primary systems fused *at least 5 subsystems*, reflecting a strong trend toward **ensemble methods** for best performance [42]. Despite their complexity,

these solutions demonstrated that the rich artifacts left by TTS and VC algorithms (e.g., slight glitches in spectral texture or phase inconsistencies) can be detected with high accuracy under clean conditions.

- **Physical Access (PA) Results:** The PA task was inherently more difficult due to the variability of replay conditions, but top teams still made dramatic progress. The best system for PA (team *T28*) achieved roughly **0.39% EER** and a min t-DCF around 0.14 [54]. This is an extremely low error rate, indicating that even replay attacks – which involve real acoustic propagation – can be almost perfectly detected in the controlled scenario *when the training and test conditions are similar*. However, PA baseline performance was much worse (the baseline PA EER was on the order of 13–15%, significantly higher than in LA), so the challenge for PA was closing a much larger gap [55]. The top PA submissions did reduce error rates by over an order of magnitude relative to baseline; for instance, team T28's system outperformed the baseline B01 by **58% relative** in min t-DCF [55]. Interestingly, the PA top-5 systems all had some different characteristics compared to LA: **every one of the top PA systems used neural networks with spectral (not cepstral) features, and none relied on GMMs** [55]. This suggests that detailed time-frequency representations (e.g., high-resolution STFT or filter bank outputs) plus CNN models were particularly effective for replay detection, likely because they can capture fine cues like reverberation tails or subtle frequency responses of speakers/microphones which a simple cepstral average might obscure [55]. The top PA approaches included models using **log power spectrograms, mel-spectrograms, and even raw waveform input**, often augmented with techniques like **Squeeze-Excitation layers, data augmentation, and ensembles**. For example, one top system (T28) used a modified ResNet with squeeze-excitation, feeding it with a combination of **magnitude and group delay spectra** to leverage phase information [55]. Another (T45 in PA context) used **CQT-based spectrogram features and an LCNN** classifier [55]. Most fused multiple subsystems (the winning PA primary fused 3 ResNet-based models) [56] [57], though interestingly the gap between single and primary systems was smaller in PA (suggesting one strong model could do well alone) [58]. Overall, PA performance was impressive under the evaluation's simulated conditions – it proved that even replay attacks can be effectively countered by modern CMs when the range of attack conditions is known *a priori* (as was the case in the simulation).

One caveat emerged: an analysis after the challenge found that some PA systems **overfit to the simulation specifics** – e.g., they might have learned to detect the finite set of impulse responses used for reverberation or the specific distortion of the simulation model [59] [60]. Thus, while PA 2019 results were stellar in the closed conditions, the generalization to *real, unconstrained replay* attacks was still an open problem – a problem the 2021 edition would tackle by introducing real replay data.

In summary, ASVspoof 2019 demonstrated that with expertly engineered features, deep models, and ensembling, it is possible to achieve **near-perfect spoofing detection (EER $\ll$ 1%) in lab conditions** [44]. The challenge also underscored the importance of generalization: systems had to handle unknown attack algorithms in LA, which many top models managed via robust feature learning (some unknown attacks like neural TTS still caused slightly higher errors, but overall performance was good across all A07–A19 attacks) [61] [62]. The findings and top techniques from 2019 laid the groundwork for the **more realism-focused ASVspoof 2021**.

ASVspoof 2021 Challenge Overview

New Tasks and Dataset in 2021

The **ASVspoof 2021** edition expanded the scope of spoofing detection and pushed systems closer to real-world conditions [63] [64]. It featured **three sub-challenges**: Logical Access (LA), Physical Access (PA), and an all-new **DeepFake (DF)** task [65] [66]. While LA and PA were carried over (with modifications) from 2019, the addition of DF addressed scenarios beyond the traditional ASV context.

- **Logical Access (LA) 2021:** The LA task in 2021 had the same fundamental definition (detect TTS/VC attacks injected remotely) but introduced **transmission channel effects** to mimic telephone network scenarios [67] [68]. The evaluation data for LA was no longer pristine audio; instead, the organizers took the *2019 LA evaluation set* (which contains high-quality spoofed and bona fide utterances from VCTK, using attacks A07–A19 [69]) and passed each utterance through one of several telephony transmission simulations [69] [70]. Specifically, seven conditions (LA-C1 through LA-C7) were defined, covering a mix of legacy and modern voice codecs and channels [71]:

- *LA-C1:* No channel (the reference – identical to 2019 condition, 16 kHz lossless) [71].

- *LA-C2:* 8 kHz *A-law* PCM codec over a Voice-over-IP (VoIP) connection (typical of digital telephony, 64 kbps) [71] [72].
- *LA-C3:* An **unknown PSTN** scenario – audio passed through a public switched telephone network (PSTN) from a mobile phone in one country to a VoIP gateway elsewhere, involving unspecified intermediate codecs/transcodings [73]. (This condition was deliberately uncontrolled: calls were made through actual telephone lines introducing whatever network effects happen in practice, giving a "wild" condition with unknown codec combos.)
- *LA-C4:* 16 kHz *G.722* wideband codec over VoIP (64 kbps) [74] [72].
- *LA-C5:* 8 kHz *μ-law* PCM over VoIP (another standard 64 kbps codec) [75] [72].
- *LA-C6:* 8 kHz *GSM* AMR codec over VoIP (simulating cellular voice at ~13 kbps) [76] [77].
- *LA-C7:* 16 kHz *OPUS* codec over VoIP (modern streaming codec at ~16 kbps VBR) [76] [77].

These represent a mix of high-quality and low-quality channels. Each evaluation utterance was transmitted through one of these conditions (with C1 being the no-transmission case for reference) [72] [78]. The **progress subset** (a portion of eval data released for intermediate scoring) included only a few conditions (C1–C4), while the full evaluation set included all seven, meaning some channel types (C5–C7, like GSM and OPUS) were **unseen during development** to test generalization [79] [80]. Importantly, **no new spoof algorithms** were introduced in LA 2021 – they used the same 13 attacks (A07–A19) from 2019 [69] [70]. What changed was the presence of compressions and packet losses which can obscure or distort the spoof artifacts. The **training data for LA 2021 was exactly the 2019 LA train+dev (clean audio)** [32]. Participants had to train on clean data and ensure their models would still detect spoofs after significant channel degradation. This was a step towards realistic deployments, since real-world ASV (like voice biometric login over phone) would involve noise, codecs, and variable bandwidth [81] [82]. By comparing LA 2019 vs 2021 performance, one could assess how robust countermeasures are to common telecom effects. The LA 2021 task essentially asked: can a spoof detector trained on clean speech still recognize fake audio after it's

passed through phone lines, compressed to 8 kHz, or even re-transcoded multiple times? This robustness to **channel variability** was a primary objective for 2021 [83] [32].

- **Physical Access (PA) 2021:** The PA task in 2021 also upped the realism by incorporating **actual replay recordings** in real acoustic environments [84] [59]. In 2019, all replay attacks were simulated with software. Post-challenge analysis showed that some CMs overly tuned to simulation quirks performed poorly on real replay data [59]. So for 2021, the organizers collected a new PA evaluation set where bona fide and spoofed audio were *replayed and re-recorded in physical rooms* (with a controlled setup) [85] [86]. They still leveraged simulation to have a large variety of conditions, but crucially, some proportion of trials are genuine acoustic recordings. The 2021 PA evaluation involved multiple **real room environments** (spanning a range of room sizes from small to large), multiple **microphones** (MEMS and high-quality mics), and varied **distances and angles** for both talker and attacker devices [85] [87]. For example, the setup described in the evaluation plan involved 9 different physical room configurations grouped into 3 room size categories, with the talker (bona fide source) and attacker (replay speaker) placed at various distances (D_s and D_a) and angles within those rooms [85] [88]. Multiple playback devices were used (of different qualities), and multiple recording mics captured the audio [85] [87]. In effect, the eval data covered many combinations of factors like: room size (small/medium/large), talker-to-mic distance (near/far), attacker device quality (high/low), attacker mic for recording (to capture original) and playback loudspeaker type. Because enumerating all combinations would be enormous, a subset was selected, and certain combinations were held out entirely from the progress set to appear only in the final eval (again to test generalization) [89] [90].

The **training data for PA 2021 remained the 2019 PA simulated set** [84] [59]. So participants trained on simulated replays (27 env configs and 9 attack configs as before) and then had to detect *real replay recordings* at eval time. This mismatch made PA 2021 especially challenging. It was essentially a domain adaptation problem: from simulation domain to real domain. The goal was to encourage countermeasures that don't rely on artifacts of a simulator (like perfect consistency of certain impulse responses) but rather pick up on more general replay signs (e.g., the added reverberation or frequency loss compared to an original) [59] [60]. The **evaluation protocol** still used t-DCF and EER, but because of the domain shift, one expected higher error rates than the near-zero values seen in 2019 PA. Indeed, as results later showed, PA 2021 turned out to be the hardest task, with many systems struggling on unseen real replay configurations [60]. This reflects reality: replay attacks "in the wild" can be very diverse (think of any environment from an office to a car interior, any recording device from a pro mic to a smartphone, etc.), so building a single robust detector remains difficult.

- **Speech Deepfake (DF) 2021:** The new **deepfake task** was introduced to broaden ASVspoof beyond just securing ASV systems, tackling the problem of detecting AI-generated speech *in general media* [91] [92]. In the DF scenario, there is no speaker verification system at play; it's a **standalone deepfake audio detection** (sometimes called *fake audio*, *synthetic speech detection*, or *audio deepfake detection*). The use case is, for example, identifying if an audio clip of a celebrity's speech online was actually synthesized to spread misinformation [64] [93]. The DF task is similar to LA in that the spoof attacks are TTS or voice conversion, but with two key differences: (1) The audio might come from various sources and **be post-processed by media codecs or compression** (since deepfakes often circulate in compressed form on the internet) [63] [93]. (2) The focus is not on protecting an ASV system, so the metric is back to plain detection EER (no t-DCF) [94] [95]. The DF 2021 dataset was by far the most diverse. It *partly* reused some 2019 data (the same VCTK-based spoofed utterances A07–A19 from LA 2019 for a portion of trials) [96], but it also incorporated **two new source corpora**:

the audio from the **Voice Conversion Challenge (VCC) 2018 and 2020** [96] [97]. The VCCs are international competitions where many teams submit converted speech using a variety of algorithms. By including VCC data, the DF evaluation had a *huge variety* of spoofing methods: **over 100 different TTS/VC systems** contributed by participants of VCC 2018, VCC 2020, and ASVspoof 2019 [98] [99]. These encompassed a wide range of techniques: traditional parametric vocoders, waveform concatenation, modern neural auto-regressive models, non-autoregressive models, etc., often with unknown details (some were open-source, some proprietary) [99] [100]. In fact, the organizers grouped these attacks into **five broad "vocoder" categories** (traditional, neural autoregressive, neural non-autoregressive, unit selection/concatenative, and "unknown") for analysis [99] [100]. Unlike LA, where the attacks had specific IDs, the DF attacks were so numerous that generalization across *entirely new spoof types* was a central challenge. The **bona fide data** for DF also came from multiple sources: VCTK (from ASVspoof) plus VCC's datasets which in turn are from **DAPS and EMIME corpora** (different recording conditions, languages) – yielding **93 different speakers** (50 female, 43 male in eval) which is more than LA/PA [101] [102]. This was to prevent overfitting to peculiarities of one dataset's speech. Finally, to simulate real-world sharing, every audio (real or fake) in DF was **encoded and decoded with lossy codecs** [103] [104]. They defined **nine DF conditions (DF-C1 to C9)** covering common media codecs and even double-compression scenarios [105] [106]:

- DF-C1: no compression (16-bit 16 kHz PCM, just downsampled from source) [105].

- DF-C2: Low bitrate MP3 (≈80-120 kbps VBR) [105].
- DF-C3: High bitrate MP3 (≈220-260 kbps) [105].
- DF-C4: Low bitrate AAC (m4a container, ≈20-32 kbps) [106].
- DF-C5: High bitrate AAC (≈96-112 kbps) [106].
- DF-C6: Low bitrate Ogg Vorbis (≈80-96 kbps) [107].
- DF-C7: High bitrate Ogg Vorbis (≈256-320 kbps) [107].
- DF-C8: **Double compression**: first MP3 then AAC (simulate a file uploaded in one format then re-downloaded/recompressed in another) [108].
- DF-C9: **Double compression**: first Ogg then AAC [108].

Some of these (C8, C9) were only present in eval not in the progress subset, again to test how systems handle totally novel conditions [105] [109]. The DF progress set had fewer codecs (likely up to C6 only). The design of the DF task hence stressed **cross-domain generalization**: a detector needed to handle different source datasets (train on 2019 VCTK perhaps, but face VCC and others), different spoof algorithms (many never seen in training), and different channel codecs (again, possibly not seen in training). It's a nearly open-set problem, reflecting the real challenge of detecting *new* deepfake methods under compression. For DF, **no specific training set was provided** aside from recommending use of ASVspoof 2019 LA train (clean) or one could use the progress data with caution [32] [96]. Most participants thus trained on the 2019 data (which has only 6 known attacks) augmented with their own simulations of codecs, etc., or used any allowed techniques to bridge the gap. The DF metric was simply EER, since there's no concept of an ASV system to protect [94] [110].

In summary, **ASVspoof 2021's databases** were far more varied and challenging than 2019's. The number of trials also ballooned (e.g., DF eval had ~534k spoof trials, PA eval ~627k spoof trials [111], which indicates multiple spoof instances per original, especially from all those VCC attacks). The **objectives** were clear: (1) test robustness to transmission/channel effects (LA), (2) test robustness to real acoustic playback (PA), (3) test generalization to unseen attacks and compression (DF) [83]. This evolution from 2019 to 2021 marks a

shift from “Can we detect spoofing in ideal settings?” to “Can we detect spoofing in the wild, where fake audio might be low-quality or differently distributed?” [112] [64].

Evaluation and Metrics in 2021

The evaluation procedure in 2021 was similar to 2019 with some additions. **LA and PA tasks** retained **min t-DCF as the primary metric** (with a fixed ASV system and costs identical to 2019) [94]. **DF task** used **EER% as the primary metric** [92]. Participants had to submit scores for a *progress phase* and a *final evaluation phase*. The progress phase allowed teams to get feedback on a subset of eval data (without knowing ground truth) – essentially it was an online leader-board to validate generalization before final submission [113] [114]. Unlike a typical dev set, the progress set did not include speaker labels or bona fide/spoof labels for tuning; participants just got metric results from the server. This was done to encourage blind generalization rather than overfitting to known dev labels [114]. The final rankings were based on the evaluation subset results.

No new training data was provided for any task [32] [115]. Teams were explicitly encouraged to leverage **data augmentation** to compensate for the domain mismatches [116] [32]. For example, adding noise, reverberation, or codec compression to training data could simulate some of the unknown conditions in LA/PA/DF. External data rules were slightly relaxed in that using non-speech resources (e.g., RIRs, noise backgrounds, codec libraries) was allowed and even promoted [32] [33]. But using external *speech* (like other speech corpora or other spoof datasets) was still generally not permitted, to keep the playing field level [32] [33]. The idea was that a strong solution should handle these variations through clever use of the given data and augmentations, not by simply training on a huge corpus of similar data.

The evaluation metrics were computed similarly to 2019. For LA/PA, min t-DCF was normalized such that an unprotected ASV system corresponds to 1.0 and a perfect system to 0 (this normalization was same as 2019’s “normalized min t-DCF”) [117] [118]. The organizers also reported the pooled EER for reference. For DF, just EER (as a percentage) was reported.

Baseline Systems in 2021

ASVspoof 2021 provided an upgraded suite of baseline systems, reflecting advances since 2019 [119] [120]. Four baseline CMs were released (with reproducible code on GitHub [121] [119]):

- **B01: CQCC-GMM (baseline)** – Same as 2019’s B01, included for continuity [120].
- **B02: LFCC-GMM** – Same as 2019’s B02 [120].
- **B03: LFCC-LCNN-LSTM** – A new baseline using deep learning. It extracted LFCC features but fed them into a Light CNN followed by an LSTM (recurrent layer) to capture temporal dynamics [119] [120]. This model, denoted in results as B03, introduced learnable feature processing and sequence modeling, expected to outperform the static GMM.
- **B04: RawNet2** – Another new baseline, **RawNet2**, which is an end-to-end model operating directly on the raw audio waveform [119] [122]. RawNet2 (from Jung *et al.*, 2020) employs learned sinc filters in its first layer and a deep convolutional architecture to produce embeddings from the raw signal, specifically designed for anti-spoofing [123]. Using raw input can capture nuances lost in feature extraction, and RawNet2 was among the state-of-the-art single systems circa 2020.

The baseline results on the 2021 tasks illustrated the **performance gap between classic and deep approaches**. For instance, on the LA eval set with channel effects, the old CQCC-GMM (B01) had around

15.6% EER (normalized min t-DCF ~0.50) whereas the RawNet2 baseline (B04) achieved about **9.5% EER** (min t-DCF ~0.426) [118]. Similarly, the LFCC-LCNN baseline (B03) was in the ~9.3% EER range, much better than the 19.3% EER of LFCC-GMM (B02) on LA [118]. This confirmed that neural network approaches significantly improve generalization to channel effects compared to GMMs. On the PA task, all baselines performed worse (since real replay data is tough): for example, in one analysis, B01 and B02 had around 36–38% EER on PA eval, while B04 (RawNet2) still had ~34% EER [124]. And for the DF task, even the best baseline (RawNet2) had relatively high EER (around mid-20s%) meaning the task was extremely challenging (we know from final results the top systems got it down to ~15% EER, so baseline was much higher) [125]. The baseline code gave participants a starting point and highlighted that **data augmentation would be essential** – e.g., the baseline code results posted on Codalab included versions "with non-speech augmentation" which slightly improved performance by trimming non-speech regions etc. [126] [118].

In short, 2021's baselines demonstrated that *even simple deep models outperform GMMs by a large margin under mismatched conditions*, validating the community's shift toward deep learning-based countermeasures. RawNet2's inclusion signaled that end-to-end learning was a viable path, though still not a silver bullet for issues like heavy compression.

Results and Comparison of 2019 vs 2021

The ASVspoof 2021 challenge had dozens of participating teams (even more than 2019, with 50+ teams submitting to at least one track). The results were revealing in several ways:

- **LA 2019 vs LA 2021:** Not surprisingly, performance on the LA task **degraded under channel effects**. Whereas 2019 LA saw multiple systems with <1% EER, in 2021 LA the top systems had higher error (because detecting fakes in 8 kHz noisy audio is harder). Nonetheless, the best 2021 LA systems still achieved very low min t-DCF and EER. In the evaluation results, many teams came close to the ASV floor for LA progress (clean-ish conditions), and only a small cost increase for the full eval with channels [127] [128]. The **top LA 2021 system** (primary) had a min t-DCF around 0.12 and EER ~2–3% on eval (this can be inferred from plots; exact numbers in literature show best LA EER around 2.5% for eval) – a bit higher than 0.22% from 2019, but given the channels, still very good. The majority of LA submissions in 2021 significantly outperformed the B01/B02 baselines, even under unknown channel conditions [129] [117]. A noteworthy observation was that **channel variation did cause detection errors** (e.g., narrowband 8 kHz conditions yielded worse performance than wideband) [130] [131], but many top systems coped by applying their own front-end normalization or multi-condition training. According to the overview paper, *most LA systems used some form of channel simulation augmentation in training*, which helped close the gap between conditions [127] [132]. Top-5 LA systems *all* were ensembles of several models, and each used different augmentation strategies (no single augmentation method was common to all) [127] [132]. For example, some added noise or RIRs, some trained on codec-transcoded data, etc. The diversity indicates that tackling channel effects can be done in many ways (front-end filtering, back-end adaptation, etc.), and teams explored all avenues. In terms of features, **short-term spectral features remained dominant** (just as in 2019) – mel-spec, CQT, LFCC were common inputs – often after some pre-processing like dereverberation or bandwidth extension [127] [132]. In sum, LA 2021 showed that **with sufficient augmentation, countermeasures can remain highly effective even over telephony**. The best systems achieved normalized min t-DCF values not far above the ASV floor, meaning only a minor impact of the channel on overall ASV+CM performance [127] [128].

- **PA 2019 vs PA 2021:** The PA 2021 task turned out to be **the most challenging of the three** in 2021 [60] [133]. This is because the training was on simulated data, but evaluation included real replay captures, introducing a domain shift. The result was that even the best PA systems had a noticeable performance drop compared to 2019's near-perfect scores. Many teams that did well in LA and DF struggled to achieve low t-DCF for PA. The *top PA 2021 system* had a min t-DCF around 0.24 (normalized) and an EER of perhaps ~9–10% (reading from challenge results charts) – far higher than the 0.4% EER of 2019's best PA. Indeed, the overview notes that all systems' performance was "substantially worse than the ASV floor" in PA, and none got close to perfect [117] [134]. Still, there was progress: the best systems did beat the baselines by a good margin. And importantly, a number of teams outperformed the B01 baseline even though B01 had an advantage of being tuned to simulation (some learned to generalize). Techniques that helped included heavy use of **Room Impulse Response (RIR) augmentation** (simulating many different reverb conditions in training) and combining models that specialize in different aspects (e.g., one model might focus on detecting slight microphone frequency responses, another on noise patterns) [135] [136]. According to the top systems summary, most PA top-5 systems were *also ensembles and also used augmentation*, similar to LA [135] [136]. Some teams employed *domain adaptation* tricks, like fine-tuning on some real data (progress set) or using unsupervised feature alignment, but since no real labels were given, this was tricky. Overall, PA 2021 highlighted that **mismatch between train and test (simulation vs reality) is a critical problem** – one that wasn't fully solved by 2021. It likely motivated further research into techniques like domain generalization and one-class learning for replay. In comparing with 2019, where PA was "easy" in the closed set, 2021 PA demonstrated a more realistic barometer of replay detection under unforeseen conditions.

- **DeepFake (DF) 2021 results:** The DF task was essentially a new challenge altogether, and it proved to be **very difficult**. On the *progress subset* (which had fewer codec conditions and perhaps easier examples), teams showed good performance – 23 of 33 systems had <10% EER on progress, and the best was <1% EER [137] [125]. This gave an initial impression that deepfake detection might be solved. However, the *evaluation subset* – with more codecs and entirely new spoofing methods – caused error rates to skyrocket. Every single submitted system had an **EER above 15% on the evaluation set** [137] [125]. The best systems were around 15-20% EER, and even though that significantly beat the baseline RawNet2 (which was ~25-30% EER, since 18 teams outperformed B04) [138], it's still a substantial error rate in absolute terms. In practical terms, this means even the top deepfake detectors would miss or confuse one out of five fake clips – not yet reliable enough for high-stakes use without further improvement. The spread of results indicated that **some teams generalized much better than others**: a few clusters of submissions were near the 15-18% EER range, while many others were far worse (some above 30% EER). The top-5 DF systems were all **ensemble models with extensive augmentation** [139] [140]. One interesting note is that one of the top DF submissions was *not* an ensemble (the only single-model among them), yet it performed competitively – this turned out to be a graph network-based system (related to AASIST, discussed later) that inherently captures a lot of information without needing ensemble [139] [140]. Common techniques among DF systems included training on as many codec conditions as possible (some teams pre-compressed their training data with random codecs each epoch) and using features that are somewhat codec-invariant (e.g., focusing on phase distortions or using convolutional layers that learn across full bands so that missing high-frequency from 8kHz audio doesn't break it). Still, the codecs did have measurable effects: e.g., results showed that **Ogg Vorbis compression was slightly easier for detection than AAC or MP3** (perhaps Ogg preserves more telltale cues) [141] [130]. Also, **double-compressed audio (C8, C9)**, which one might expect to be hardest, yielded only moderately

higher EER than single-compressed, indicating that once you handle one compression, a second doesn't fully confound the system [130] [142]. The bigger impact was the source domain: audio from VCC 2020, for example, had higher EER than audio from VCC 2018 or ASVspoof 2019 [143] [144], suggesting that completely new attack algorithms in 2020 challenged the detectors.

In comparing DF to LA: LA can be seen as a simpler special case of DF (same types of attacks, but all from one database and no compression). Indeed, LA results were much better than DF. This emphasizes that **heterogeneity is a major obstacle** – when the fakes come from a distribution very different from what a system has seen, performance drops. ASVspoof 2021 DF track thus served as a wake-up call: even state-of-the-art fake audio detectors can struggle in open-concept evaluations. It aligns with observations in the vision deepfake field (face deepfakes) where detection is easier when test fakes are similar to training fakes, and much harder when they're completely novel.

Summary of technique trends (2021): According to the organizers' analysis, virtually all top systems across LA, PA, DF used **data augmentation** in some form, but there was *no single universally best augmentation* – teams tried various approaches like additive noise, codec simulations, speed perturbation, etc., often in combination [127] [132]. Another commonality was the use of **ensemble systems**: out of the top-5 of each task, nearly all were ensembles of multiple subsystems (except one DF entry) [145] [146]. This indicates that fusion of different models/feature sets remained crucial to achieve robustness. The types of models used included CNNs (ResNets, EfficientNets, etc.), some RNNs or Transformer layers for temporal context, and even emerging approaches like *graph neural networks* (for example, one of the top LA/DF systems applied a graph convolution to model relationships between subbands over time) [147] [148]. Simpler classifiers like GMMs or SVMs virtually disappeared from the top ranks – deep learning dominated. Features, while mostly spectral, ranged from traditional cepstra to learned embeddings. A noteworthy development by 2021 was that some models leveraged *pre-trained* representations: for instance, a few submissions used features from *self-supervised audio models* or from *ASV models (x-vectors)* to help spoof detection, though with mixed success.

Comparing the two editions: **ASVspoof 2019 established that known forms of spoofing can be detected with high accuracy under controlled conditions**, and that both classical and deep methods could achieve good results (with deep methods showing an edge). **ASVspoof 2021 pushed the field to consider unknown conditions and attack types**, revealing that while progress has been made, generalization is far from solved (especially for DF). The inclusion of the DF task was prescient, as the last couple of years have seen rapid improvements in voice synthesis – making the need for reliable deepfake audio detectors even greater.

The evolution from 2019 to 2021 also reflects a shift in research focus: from purely maximizing accuracy on known scenarios to **improving robustness**. The challenge objectives explicitly included *"robustness to channel variability"* and *"generalisation to new domains"* [83]. As such, many 2021 papers and systems presented methods for domain adaptation, augmentation, and one model handling multiple conditions. This experience is guiding the design of the next generation of countermeasures.

Advanced Techniques in Anti-Spoofing Systems

Building on the lessons from ASVspoof, we now summarize key techniques and approaches used in modern anti-spoofing systems, including feature extraction, modeling, augmentation, and ensembling. These are

the tools that enabled top challenge performances and are recommended for anyone developing a spoofing detection system today.

Feature Extraction Methods

Careful feature engineering (or learning) is fundamental for spoof detection, because spoofed vs. real speech often differ in subtle ways that good features can highlight.

- **CQCC (Constant-Q Cepstral Coefficients):** CQCCs were a breakthrough feature introduced in 2016 for spoofing detection. They use a constant-Q transform to obtain a time-frequency representation with a **logarithmic frequency axis**, which provides higher frequency resolution at lower frequencies and vice versa [36]. After scaling, a cepstral analysis is applied (similar to MFCC calculation). CQCCs are effective at exposing artifacts of synthesis in both high and low frequency bands. They were the top features in ASVspoof 2015/2017 and remained a strong baseline in 2019 [36] [37]. One reason is that TTS or VC signals often have unnatural spectral details (like extra harmonics or noise floors) that constant-Q analysis can pick up across octaves. CQCC features (with dynamic coefficients) still serve as a solid choice for classical systems or as inputs to CNNs.

- **LFCC (Linear Frequency Cepstral Coefficients):** LFCCs are essentially standard cepstral coefficients computed on a linear-frequency filter bank [39]. They played a big role as a simple but competitive feature (the 2019 baseline B02 used LFCC). LFCCs preserve more high-frequency detail than MFCCs (which emphasize perceptual bands), and high-frequency noise or aliasing can be a giveaway of synthetic speech. Many challenge participants used LFCCs or similar spectral coefficients (sometimes with modifications like inverted MFCC or subband spectra) as inputs to their neural networks [49] [149]. LFCCs are straightforward to implement and, with delta features, provide a compact representation of the speech spectrum suitable for GMM or DNN classifiers.

- **Spectrogram-Based Features:** Rather than compressing information into cepstra, many state-of-the-art systems feed the **raw spectrogram or filtered spectrogram** into deep models. **Log-magnitude STFT spectrograms** (e.g., on 20–30 ms frames) can be directly input to CNNs, treating them like images [50] [150]. The advantage is no information loss from averaging, so the model can learn any discriminative pattern (e.g., fine harmonic structures, noise textures). Variants include **Mel-spectrograms** (spectrogram filtered into mel bands), which reduce dimensionality while retaining overall shape, and **CQT spectrograms** (constant-Q transform before taking log magnitude), which several top systems have used to capture periodic cues in voices [49] [53]. Another powerful representation is the **group delay spectrogram** or **phase spectrum**. Some systems compute the phase response's derivative (group delay) and use its magnitude as an image – useful for replay detection because phase encodes echoes and device effects. For example, team T10 in ASVspoof 2019 used group delay grams with cepstral mean normalization as input to a ResNet, yielding excellent PA performance [55]. Overall, spectrogram inputs (magnitude or phase or both) are prevalent in deep learning approaches, as they let the model decide what features are important, rather than handcrafting.

- **Raw Waveform and Learnable Front-ends:** End-to-end approaches attempt to learn features directly from waveform. Techniques like **SincNet filters** (used in RawNet2) learn bandpass filters optimized for the task [119] [123]. The network can thus focus on frequency bands that maximize spoof vs. real separation. RawNet and its improved versions use such learnable filters followed by layers

akin to CNNs to produce embeddings. The benefit is potentially discovering features humans might not consider (maybe subtle phase shifts or micro-signal patterns). However, training end-to-end can require more data to not overfit. In practice, raw-waveform models have matched but not dramatically exceeded spectrogram-based CNNs in ASVspoof so far [118]. Nevertheless, end-to-end is promising especially as more training data becomes available or through pre-training on large corpora.

- **High-Frequency & Phase Features:** Spoofing often leaves clues in high-frequency ranges (e.g., above 4 kHz) where vocoders may drop energy or introduce aliasing. Features like **Inverse MFCC (IMFCC)** specifically emphasize high-frequency content by using an inverted mel scale [52]. This was used in some fusion systems in 2019 (one subsystem might be IMFCC-based). Additionally, phase-based features: aside from group delay, some use **Magnitude/Phase Co-Occurrence** (treat phase as an image to analyze along with magnitude). Replay attacks, for example, can be detected via anomalies in the phase response introduced by loudspeakers and rooms.

- **Temporal Features and Others:** Some research has used **prosodic features** (e.g., pitch patterns, duration patterns) under the rationale that TTS might not capture human prosody perfectly. But in ASVspoof challenges, such features haven't been dominant – probably because current TTS can model prosody fairly well or because the countermeasures focus on short-term artifacts. There's also the approach of **feature learning**: using an autoencoder or unsupervised model to learn an embedding of speech that highlights synthetic aspects. For example, one could train a denoising autoencoder on real speech and see if it reconstructs spoofed speech poorly (thus creating a feature for detection). Some top systems implicitly do this by training on one-class (bona fide) and using reconstruction error for detection, but this is still emerging.

In conclusion, **multi-feature fusion** is powerful – combining cepstral features that summarize the spectrum with detailed spectrogram patches and with phase info can give a holistic view. The 2019 winning ensemble literally combined subsystems operating on seven different feature types [43]. As a practitioner, one might start with LFCC or mel-spectrogram as a baseline feature and later incorporate additional features if performance plateaus.

Deep Learning Models

Deep neural networks are now at the core of anti-spoofing systems, providing superior modeling of complex feature distributions:

- **Convolutional Neural Networks (CNNs):** CNNs excel at learning local patterns in spectrograms that distinguish spoof vs. real. Many architectures have been explored. The **Light CNN (LCNN)** with Max-Feature-Map activation was popular around 2019 [51] – it's efficient and was designed for face anti-spoofing originally, performing well for speech too. **ResNets** have been widely used; e.g., ResNet18, ResNet34 with modifications like **Squeeze-and-Excitation (SE) blocks** were top performers (SE blocks help the network focus on important channels/frequencies) [55]. There were challenge entries with **DenseNets**, **MobileNets** (T05 used MobileNetV2 in several subsystems) [48], and even custom CNNs for specific features (like a shallow CNN for phase spectra). CNNs can also be used as front-ends to produce embeddings which then go into an LSTM or a fully-connected classifier. For example, some systems did: spectrogram -> CNN -> embedding -> linear classifier for final decision, with a softmax or logistic output. We also saw **multi-branch CNNs** in some solutions: e.g., one

branch processes low frequencies, another processes high frequencies, then merge. This can allow specialized learning (since replay might mostly manifest in high-freq drop-off, while TTS artifacts might be broadband). All these variants show the flexibility of CNNs for the task.

- **Recurrent Neural Networks (RNNs) and Temporal Models:** While CNNs capture frame-level patterns, RNNs capture sequence patterns. Spoofing cues might not be just static – e.g., certain anomalies might only appear at phoneme transitions or in long-term energy contours. Some challenge submissions included **LSTM** or **GRU** layers on top of CNNs (the baseline B03 is an LCNN + LSTM) [120]. These can model how features evolve over time. For instance, a genuine speech might have natural variations in pitch and timing that a VC-converted speech does not replicate perfectly; an LSTM could theoretically pick that up. However, pure RNN-based systems (like using only an LSTM on MFCC sequences) are less common now, as CNNs with global pooling can also capture some temporal info. A popular approach is **CRNN** (Convolutional-Recurrent NN): first few layers are CNN (to reduce dimensionality and extract local features), then one or two bidirectional LSTMs, then fully connected. This was used by some 2019 teams and can improve performance at cost of complexity. Another form of temporal modeling is using **attention mechanisms** – e.g., a model can attend to the most relevant frames (maybe the noisiest frames for spoof). A few 2021 systems likely used an attention pooling instead of simple averaging.

- **Graph Neural Networks (GNNs):** An exciting recent development is the use of graph-based deep models for spoof detection. The idea, as demonstrated by AASIST (2021), is to treat the time-frequency representation as a graph: each time frame or each frequency bin is a node, and edges connect nodes to capture local relations [147]. A **Graph Attention Network (GAT)** can then learn patterns jointly in time and frequency domain beyond the rigid grid of CNN filters [147]. AASIST specifically builds two graphs – one along the time axis, one along frequency – and processes them to detect anomalies. Graph networks can capture complex dependencies (for example, if a certain frequency band always behaves oddly whenever another band has a spike, which could be a telltale signature of a vocoder). In initial results, AASIST achieved SOTA on ASVspoof 2019 and also did very well on 2021 data, all without needing an ensemble [139]. This points to a direction of making single-model solutions more powerful by architectural innovation. It's a bit advanced for someone starting out, but open implementations exist and they show that GNNs can outperform traditional CNNs for this task.

- **Ensemble and Fusion Techniques:** Rather than a specific model type, ensemble is a strategy to combine models. As evidenced many times, ensembles yield significant gains [42] [145]. Fusion can be done by **score averaging**, **weighted voting**, or by training a secondary classifier (like a logistic regression or shallow MLP) on the concatenated scores of subsystems. For example, the T05 system fused 7 subsystems by normalizing and equal-weight averaging their scores [49] [50]. Some others use **majority voting** if they treat each sub-model's decision discretely. A key to successful ensembles is diversity: combining models that make different errors (like one model good at catching replay, another at catching TTS). In practice, people ensemble models trained on different features or with different architectures, or even the same architecture trained on different subsets (bagging). A lesson from ASVspoof is that a well-fused ensemble can reduce error dramatically – e.g., T05 primary (fusion) had 56% lower t-DCF than its best single component T45 [42]. However, increasing ensemble size hits diminishing returns after a point (and increases runtime). There's interest in creating a *single model as effective as an ensemble* (AASIST's motivation [151]), which is better for deployment. But currently, top performance still often involves 3-5 models fused.

- **Calibration and Loss Functions:** Most systems use a binary classification loss (like cross-entropy or focal loss) to train on real vs spoof. Some 2019 winners used **Angular Softmax (A-Softmax)** or similar losses that enforce a larger margin between classes in embedding space [51]. This helps if one wants the embeddings to be highly discriminative and robust. The t-DCF metric itself wasn't directly used as a training objective (since it's non-differentiable), but some teams tried to approximate it or at least monitor it during training. Post-training, **score calibration** may be applied (especially if one wants to tune to a specific operating point). This might include scaling and shifting scores based on dev set statistics to minimize t-DCF.

- **Model Regularization:** Since data is limited, techniques like **mixup (interpolating between speech samples)**, **dropout**, and weight decay are common to avoid overfitting. Some unique regularization specific to spoofing includes augmentation as discussed below – e.g., using noise as a regularizer or random filtering.

In essence, the field has moved towards complex neural networks, but with a caution: generalization to unseen conditions is key. It's not enough for a model to memorize spoof artifacts in training; it should detect fundamental differences in how speech is generated or replayed. This is motivating research into *one-class learning* (learning only bona fide speech distribution and flagging anything outside it as fake) and *domain generalization* (training models that are invariant to channel, speaker, etc., focusing only on spoofing cues).

Data Augmentation and Training Strategies

If one theme stands out from ASVspoof 2021, it's the crucial role of **data augmentation** in training robust anti-spoofing models. Augmentation artificially expands the training data diversity, helping models not to overfit to specific conditions. Key augmentation techniques include:

- **Channel and Codec Augmentation:** Simulate the effects of transmission channels, microphone frequency responses, and compression on training data. For LA 2021, participants would take a clean training utterance and randomly apply a codec like G.711 (8 kHz) or a bandwidth filter to mimic telephony, then use that as an additional training sample [32] [33]. This way, the model learns to ignore distortions that are from channel, while still detecting spoof patterns. Similarly, in the DF task, applying MP3/AAC compression with random bitrates to both bona fide and spoof during training was common. One can even stack codecs (first compress with one, then with another) to simulate the double-compression cases. Tools like `ffmpeg` or codec APIs can generate these on the fly. This approach was effectively necessary to approach the DF task – models trained only on clean fakes struggled when faced with compressed fakes [125].

- **Noise and Reverberation:** Adding background noise or reverberation (via convolution with room impulse responses) makes the model robust to environments. While ASVspoof 2021 did not explicitly add noise in evaluation, real recordings naturally include some noise and reverberation. Many teams augmented PA training data with diverse RIRs (some used public databases of impulse responses to simulate echo in various rooms) [59] [86]. This helps the model learn replay cues that persist across different reverbs, rather than latching onto one particular synthetic reverb signature. Similarly, adding a bit of noise can make the model pay attention to spoof artifacts rather than silent periods or DC offset, etc. The evaluation plan even allowed using noise samples and impulse responses from

external sources [33]. For a practical system, one should train on a mixture of clean and noisy conditions, possibly using noise from e.g. MUSAN or Audioset as background.

- **Speed and Pitch Perturbation:** Slightly speeding up or slowing down the audio (e.g., ±5%) or shifting its pitch can augment the data. This is common in speech recognition training; for spoofing, it can help because it creates variations of the same utterance with different prosody. An interesting aspect: voice conversion often doesn't preserve natural pitch fluctuations perfectly, so training the model on pitch-augmented real speech might teach it a range of what "normal" pitch variation looks like, making it easier to spot when a spoof lacks that natural variation. Some teams also randomly changed formant frequencies (vocal tract length perturbation) to diversify speakers. These techniques force the model to not rely on specific speaker identities or exact durations, focusing instead on source characteristics.

- **SpecAugment and Time-Frequency Masking:** Borrowed from ASR, SpecAugment involves masking out random blocks of time frames or frequency bins in the spectrogram during training. By doing this, the model can't overly rely on any one frequency region or any brief moment in time – it has to glean cues from the remaining parts. This can improve generalization and robustness. For spoof detection, one risk is that important artifacts might get masked; however, using small masks typically regularizes the network without removing all artifact info. Some participants likely used this (though it's not always reported).

- **Adversarial Training:** A more advanced augmentation is to generate adversarial examples – inputs that are subtly modified to fool the model – and train on them to strengthen the model. There's research on adversarial attacks in speaker recognition and spoofing; a robust model might incorporate adversarial training to not be easily spoofed by an attacker who knows the model. This is beyond what challenge participants did, but it's a consideration for future.

- **Balanced Sampling and Hard Example Mining:** The training datasets are often imbalanced (especially in eval, spoofs outnumber bona fide by a lot). During training, one can oversample the minority (bona fide) or use balanced batch sampling (ensuring each batch has equal bona fide and spoof). This prevents the model from getting biased towards the majority class. Also, monitoring which samples are hardest (highest loss) and focusing on them can improve learning. Some teams probably did something like focal loss (which downweights easy examples) to focus on challenging cases (like those unknown attacks A07–A19 for LA).

In practice, a good recipe might be: take all your training clips, and randomly apply one or more augmentations each time you feed them to the network: e.g., with 50% chance add a random RIR, with 50% chance add noise at SNR sampled between 10 and 30 dB, randomly lowpass or bandlimit some, randomly time-stretch by ±3%, etc. The PyTorch `torchaudio` and `nn.AudioEffects` or specialized libraries can help. Also, the ASVspoof organizers provided some augmentation code as part of baseline (for example, one of the baselines had a switch to add impulse response augmentation).

Augmentation proved vital in 2021: systems that did not augment were easily identified by their poorer generalization (the overview paper notes that the teams who didn't augment fell behind) [127] [132]. It essentially became a necessity to simulate the diversity of eval conditions.

Performance Metrics and Evaluation

We've touched on metrics earlier, but here we summarize how to measure performance in anti-spoofing, as that guides model development:

- **Equal Error Rate (EER):** This remains the go-to metric for stand-alone spoof detection performance [152] [153]. It's easy to interpret (the lower, the better; 0% means perfect). It doesn't require setting a decision threshold – it's the point on the DET curve where miss and false alarm are equal. In research papers and competitions (especially the DF task), EER is widely reported. Typically, one computes scores for all trials (with higher scores meaning "more likely spoof" in a common convention) and finds the threshold where false rejection = false acceptance. EER was the official metric for ASVspoof's deepfake track [92] and is also used in related challenges (e.g., the ADD 2022 challenge for audio deepfakes).

- **min t-DCF:** When the spoof detector is used in tandem with an ASV system, the **t-DCF** metric is more informative [26] [94]. It accounts for three error types: false reject of real (user inconvenience), false accept of imposter (ASV error), and false accept of spoof (spoof breakthrough). The ASVspoof version of t-DCF assumes fixed costs and priors (like how likely spoofs are, etc.) and a fixed ASV performance (they took an ASV system with a certain EER, around 1-2%). The output is a normalized cost where 0 = no cost (ideal) and 1 = cost as bad as not having a detector [26]. For instance, a min t-DCF of 0.5 means the system halves the potential loss from spoof attacks compared to having no detector. In 2019, min t-DCF was the primary metric and it revealed interesting things: e.g., some systems had low EER but not as low t-DCF because they might have rejected some real users or certain attacks were particularly harmful [45]. Using t-DCF encourages designers to consider not just overall accuracy but how the detector works with ASV. For example, if an ASV system is very good at rejecting zero-effort impostors but very vulnerable to spoof, you might weigh spoof false accepts more. In implementation, the **normalized min t-DCF** calculation script (provided by ASVspoof) should be used. It's a good practice to optimize your threshold for min t-DCF on a dev set rather than just EER threshold if you care about deployed performance, since the operating point that equalizes error rates might not be the cost-optimal one for protecting ASV [117] [134].

- **Detection Error Tradeoff (DET) and other analyses:** Aside from scalar metrics, researchers plot DET curves (miss vs false alarm in a log scale) to see performance across operating points [154] [155]. They also break down results by attack type or condition. For example, evaluating EER per attack algorithm (A17 vs A16 etc.) shows which spoofs are hardest [156] [54]. Or in 2021, EER per codec condition (MP3 low vs high) was analyzed [157]. A robust system should perform consistently across conditions, whereas a weakness might be revealed if one condition has much higher EER. These analyses guide improvements (e.g., if your model did poorly on a particular unknown attack, maybe you need to augment more variations similar to it).

- **Computational Efficiency:** While not a direct metric, it's worth noting the practical side: inference time and model size matter for deployment. Some challenge papers discuss complexity – e.g., T45 (light CNN) was appealing for being fast [51]. If you plan to run detection in real-time on an edge device, you may favor a smaller model (slightly higher EER but runs in 0.1x realtime) over a huge ensemble. There's ongoing work on distilling ensembles into single models or using model compression.

- **Benchmark Datasets:** When building a system outside of the challenge, one can evaluate on ASVspoof corpora which are public. ASVspoof 2019 and 2021 datasets are now accessible (for research) and provide standard benchmarks. For instance, one might report performance on ASVspoof 2019 LA eval (for reference, many papers state EER on that – e.g., a certain model got 0.5% EER [158]). Similarly, results on ASVspoof 2021 LA/PA/DF eval sets would contextualize your system among challenge entrants. The challenge organizers have published the *ASVspoof 2021 results summary* which includes baseline and top performances [118] [159], which you can use as reference points.

In summary, **use EER for general detection ability** and **t-DCF if you're specifically guarding an ASV system**. For a comprehensive evaluation, do a breakdown by attack type and condition. And be mindful of things like confidence intervals on EER if the number of trials is small (though in ASVspoof, numbers are large enough that that's less an issue).

Practical Guidance: Building an AI-Generated Audio Detection System

Finally, translating all this to practice, suppose we want to build our own system to detect AI-generated or manipulated audio (deepfakes). Here's a step-by-step high-level guide incorporating the above insights:

1. Choose and Prepare Data: A good starting dataset is **ASVspoof 2019 LA** (for basic fake vs real on clean speech) [12] [3]. It has bona fide from 20+ speakers and spoofed utterances from various TTS/VC methods – perfect for initial training. Then, to make the system robust: incorporate portions of **ASVspoof 2021 data** – for example, you can take the 2019-trained model and test it on 2021 LA evaluation data (with channel effects) to see how it does, or fine-tune on some 2021-like augmentations. If possible, also include **external** public data: the Voice Conversion Challenge 2018 & 2020 datasets (which the 2021 DF task used) are now available [96] [97] and contain many examples of converted speech (with ground truth). Another resource is the **Fake or Real (FoR) corpus** (if available), or **In-the-wild deepfake audio clips** with manual annotations (though these are scarce). At minimum, using ASVspoof corpora will give you a diverse set of attacks and real speech. Ensure you have a held-out test set that you never train on – e.g., use the official eval set as a proxy test to objectively measure your progress.

2. Feature Extraction & Preprocessing: Start by computing a reliable feature representation from the audio. Two established choices: - **LFCC features** (e.g., 20 static coefficients + deltas, total 60-dim) have been a strong baseline [39] [37]. You can compute these per frame (using a library or implementing via FFT). These can then feed into a neural network or even a GMM for a simple model. - **Log Mel-Spectrogram** – for example, 64 mel filter banks, 25 ms frames with 10 ms hop – as input to a CNN. This is simple using libraries like librosa or torchaudio. Normalize the spectrogram (either per utterance or globally to zero mean and unit var) [48]. Some systems do **per-utterance mean normalization** on features (CMS – cepstral mean subtraction) to remove channel bias [55]. - If you use raw waveform (for a RawNet2 model), ensure audio is downsampled to 16 kHz and maybe normalized in amplitude. RawNet2 will handle feature learning internally.

Additionally, perform **voice activity detection (VAD)** or trimming of silence. Leading/trailing silences can sometimes confuse the model or contribute nothing but could be picked up as artifact (e.g., a synthetic audio might have absolute digital zero segments). Some studies found that removing non-speech parts

improved performance for baselines [160]. So either use a VAD to trim, or include a feature to mark speech vs silence frames.

Data augmentation can be integrated here: e.g., define an augmentation pipeline to apply to the raw audio just before feature extraction. That way, each epoch the model sees a slightly different variant of the same file (as discussed in augmentation section). For example, using `torchaudio.transforms` to add reverberation or bandpass can be done before computing mel-spec in a PyTorch data loader.

3. Model Selection: For a first version, a straightforward model is a **CNN** on the spectrogram. For instance: - A **ResNet18** architecture (with 2D convolutions) has been proven effective [55]. You could use a pre-existing ResNet implementation, adjusting the first layer to accept single-channel spectrogram input instead of 3-channel image. - An **LCNN** (lightweight CNN) with Max-Feature-Map activation can also be implemented – but using standard ReLU-based CNNs (ResNet, etc.) may be easier to start with. - Ensure the model outputs either a single logit (for binary classification) or two scores (softmax for real vs spoof). Typically, a sigmoid or softmax cross-entropy loss is used.

If you're comfortable, you could also try the **RawNet2** model (official PyTorch code is available on GitHub [119] [123]). It's more complex but you might get away without manual feature extraction. However, training RawNet2 from scratch might require careful tuning.

Another modern choice: implement the **AASIST** model (the graph-based network) using the open-source code [161]. This model is quite advanced but has shown excellent generalization, and the code release from EURECOM can be adapted.

Start simple, then increase complexity. Even a 5-layer CNN can learn some spoof vs real differences. Then you can iterate: e.g., add an LSTM on top if you suspect temporal modeling is needed, or add attention pooling.

4. Training Strategies: Use a binary classification loss (BCE or cross-entropy). Pay attention to class imbalance: if you have many more spoof than real in training, you might weigh the loss of bona fide examples higher so the model doesn't ignore them. As you train, monitor the EER on a validation set (you can compute it by scoring the val utterances and finding threshold). Also monitor the min t-DCF if you have an ASV score simulation – but early on EER is easier to compute. Utilize **augmentation** heavily (as described). Start with moderate augmentations to not confuse the model early. For instance, first train a few epochs on clean data to let it latch onto basic cues, then progressively ramp up augmentation strength.

Use techniques like **early stopping** on val EER to avoid overfitting. Also, possibly use **learning rate scheduling** – since data is limited, a high learning rate might overfit quickly; consider a relatively lower LR (e.g., 1e-4 for Adam optimizer) and see how loss goes.

One useful trick: because spoof detection can be viewed as outlier detection in some ways, sometimes training the model with **one-class tricks** helps. For example, train an autoencoder on bona fide and see if reconstruction error is higher for spoof – but implementing that is a research project in itself. For a practical system, supervised binary training is usually fine given the datasets we have.

5. Evaluation and Tuning: After training, evaluate on a test set (e.g., ASVspoof 2019 eval or 2021 eval). Calculate the **EER**. Use the official scripts for t-DCF if needed (available on ASVspoof website) [27] [41] – you will need the ASV scores for that; the 2019 evaluation plan provides a set of ASV scores for the evaluation trials to compute t-DCF [28]. If your focus is general deepfake detection (not tied to ASV), EER and maybe **ROC curves** or **DET curves** suffice.

If results are not satisfactory, analyze error patterns. For instance, check if a particular attack type in eval is always missed – maybe your model never saw anything like it. You might then augment some similar data or add a subsystem specialized for that. Check if all false alarms are coming from certain speakers or codecs – that could indicate an over-sensitivity to a channel effect rather than the spoof itself, implying you need more channel augmentation or normalization.

Adjust the decision threshold according to the application: if missing a fake is worse than false alarming on a real, you might set the threshold to a low false negative rate point. t-DCF inherently accounts for such cost trade-offs, but if not using t-DCF, you manually decide threshold as needed.

6. Useful Tools and Libraries: Developing such systems is facilitated by: - **Kaldi or SIDEKIT**: Kaldi (speech recognition toolkit) has recipes for ASVspoof (for example, to compute CQCC features, train GMMs, etc.). SIDEKIT is another toolkit that had spoofing examples. These can be useful to benchmark basic systems. - **PyTorch**: Most research code is in PyTorch. As noted, the ASVspoof organizers have a GitHub repo for 2021 baselines [121] which provides PyTorch implementations of the baselines (including RawNet2, LCNN, etc.). You can use that as a starting point and then modify architecture. - **TensorFlow**: Some might prefer TF/Keras. One can implement similar models there; just there are less readily available models from the community (most share PyTorch code). - **Librosa/torchaudio**: for feature extraction and augmentation. Librosa can do mel-spectrograms, and you can write a custom augmentation pipeline (like using `sox` effects through torchaudio). - **OpenSMILE or DSP libraries**: to compute classic features (LFCC, etc.) if you want to replicate GMM baselines or include them. - Pretrained models: The community has shared some **pre-trained anti-spoofing models**. For instance, the `asvspoof` GitHub might have weights for baseline models you can use as a sanity check. There are also research models like `RawNet2` weights trained on ASVspoof, or `AASIST` pretrained weights [161], which you could use for transfer learning or as a feature extractor (embedding network).

7. Deployment Considerations: If deploying a spoof detector in a production system (say, to filter audio submissions in an application): - **Real-time processing:** If it's a streaming scenario, ensure your model can process frames in real-time. This might favor smaller CNNs or not using extremely long input contexts. - **Threshold calibration:** Calibrate on development data to achieve the desired operating point (for example, if you want <1% false alarm rate, set threshold accordingly and know your miss rate). - **Monitoring:** Even after deployment, monitor the detector's performance, as new types of fakes might emerge. Continual learning or periodic retraining with new spoof examples is advisable (for instance, as new TTS techniques become public, add them to your training). - **Combining with ASV:** If the detector is front-end to ASV, consider **score fusion with ASV scores** for decision. Sometimes it's beneficial to combine the two scores (some solutions treat it as a joint optimization). - **User Experience:** Plan for what happens when the detector triggers. If it's high security, you might outright reject. If it's consumer-level, maybe you ask for a retry or an alternate verification method.

In terms of libraries, frameworks like **ONNX** can convert your PyTorch model for use in various environments, and there are mobile deployment options (if needed).

8. Validation with Benchmarks: It's wise to test your system on the known benchmarks. For example, report its EER on ASVspoof 2019 LA eval: if you get around 1-2% EER, that's state-of-art range [162]. If it's 5-8%, then there's room to improve (since baseline was ~8% [35]). Similarly, test on ASVspoof 2021 DF eval set (if accessible) to see how it copes with heavy compressions. Achieving ~15% EER there would put you among the top systems as of 2021 [125].

By following these guidelines – leveraging the datasets, using robust features, training advanced models with augmentation, and evaluating with proper metrics – one can build a strong AI-generated audio (deepfake) detection system that encapsulates the state-of-the-art techniques forged in the ASVspoof challenges. The field is evolving, and new challenges (ASVspoof 2022/2023 and beyond) are exploring even more aspects like replay with more realism and deepfakes in multilingual scenarios. Staying updated with the latest challenge findings will continue to inform best practices for anti-spoofing systems.

Conclusion

The ASVspoof 2019 and 2021 challenges have significantly advanced our understanding of spoofing countermeasures for speaker verification and fake audio detection. In 2019, the focus was on **high-quality fake speech** (TTS/VC) and **simulated replay** attacks, and the top systems demonstrated that with carefully engineered features (CQCC, LFCC, spectrograms) and deep learning models (CNNs with angular loss, ensembles), it was possible to achieve **near-perfect detection** in controlled settings [44] [54]. ASVspoof 2019 for the first time incorporated all major attack types and introduced the t-DCF metric, emphasizing solutions that not only detect spoofs but also maintain ASV performance [26] [27]. By contrast, ASVspoof 2021 pushed countermeasures into **more realistic scenarios**: telephony networks, real acoustic replay, and diverse deepfakes [63] [64]. This highlighted the challenges of **generalization** – error rates rose when conditions were mismatched or attacks were novel, underscoring that the problem is far from solved in the wild [60] [125]. Nonetheless, the 2021 winning systems made clever use of data augmentation and model ensembles to remain effective, achieving low t-DCF in unseen channel conditions and establishing new baselines for deepfake audio detection (with ~15% EER on a very hard task) [125].

Over the two editions, we saw an evolution from GMM-based classifiers to end-to-end neural networks, from standalone feature design to integrated feature learning (like RawNet2 and graph networks). Techniques such as **constant-Q cepstral features and linear cepstral features** proved their worth in early systems [36] [39], but ultimately **deep models** that can exploit detailed **spectrogram representations** and **phase information** became the dominant approach [55]. The use of **ensemble methods** was a recurring theme, as combining multiple subsystems yielded robustness against a wider range of attacks [42] [145]. At the same time, the community has identified that purely increasing model complexity is not enough – **data diversity and realism** are crucial. This is why 2021's allowance of using impulse responses, noises, and codec simulations was so important [32] [33]: it encouraged researchers to make their models invariant to nuisance factors and sensitive only to spoofing artifacts.

For practitioners aiming to build an AI-generated audio detection system, the key takeaways are: use a **solid dataset** (like ASVspoof) as a foundation, incorporate a **variety of features** or an expressive learned representation, train a **robust model (preferably a CNN or evolving architecture like AASIST)**, and apply **extensive augmentation** to cover expected deployment conditions. Evaluate the system thoroughly with metrics like EER and t-DCF, and compare it against challenge benchmarks to gauge its standing. Fortunately, many tools and open-source implementations exist to accelerate development – from baseline systems [119] to entire challenge corpora available on Zenodo [163] [164].

As of 2025, research is ongoing to further improve spoof detection. Areas like **speaker embedding fusion** (using ASV embeddings to help detect if the voice identity is too perfect a match, which could indicate conversion) and **continual learning** (adapting to new spoof types on the fly) are being explored. The community is also focusing on **generalized fake audio detection**, unifying approaches for detecting synthetic speech, voice clones, and even audio adversarial examples.

The progress from ASVspoof 2019 to 2021 shows a clear trajectory: *towards more realistic, more challenging, but also more practically useful anti-spoofing systems*. By steadily incorporating real-world variability and broad attack diversity, these challenges ensure that the developed countermeasures are not just overfitted tricks, but have the resilience needed for real-life deployment. Through continued community efforts, we can expect that detecting AI-generated audio will become increasingly accurate, helping safeguard voice-based technologies from malicious abuse while maintaining user trust in voice biometrics and media authenticity.

Sources: The above deep dive references the official ASVspoof 2019 and 2021 challenge summary papers and evaluation plans for detailed information on tasks, datasets, and results [3] [73] [44] [125], as well as other research contributions for specific techniques and system descriptions. These resources provide further reading for those interested in the technical intricacies of each winning system and the analytical findings (e.g., factors influencing performance) from the challenges.

---

[1] [2] [4] [5] Illustration of ASVspoof 2019 physical access (PA) scenario. Replay... | Download Scientific Diagram
https://www.researchgate.net/figure/Illustration-of-ASVspoof-2019-physical-access-PA-scenario-Replay-attacks-are-simulated_fig2_341533813

[3] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [25] [26] [27] [28] [29] [30] [31] [34] [35] [36] [37] [38] [39] [40] [41] [42] [43] [44] [45] [46] [47] [48] [49] [50] [51] [52] [53] [54] [55] [56] [57] [58] [61] [62] [149] [150] [156] [162] ASVspoof 2019: Spoofing Countermeasures for the Detection of Synthesized, Converted and Replayed Speech
https://hal.science/hal-03236124/file/ASVspoof2019_TBIOM.pdf

[6] [7] [8] [9] [22] [23] [24] [154] [155] ASVspoof 2019: A large-scale public database of synthesized, converted and replayed speech
https://arxiv.org/pdf/1911.01601

[32] [33] [59] [60] [63] [64] [69] [70] [71] [72] [73] [74] [75] [76] [77] [78] [79] [80] [81] [82] [85] [86] [87] [88] [89] [90] [93] [95] [96] [97] [98] [99] [100] [101] [102] [103] [104] [105] [106] [107] [108] [109] [110] [111] [112] [113] [114] [115] [117] [118] [120] [122] [123] [124] [125] [126] [127] [128] [129] [130] [131] [132] [133] [134] [135] [136] [137] [138] [139] [140] [141] [142] [143] [144] [145] [146] [152] [153] [157] [159] [160] arxiv.org
https://arxiv.org/pdf/2210.02437

[65] [66] ASVspoof 2021: accelerating progress in spoofed and deepfake ...
https://www.isca-archive.org/asvspoof_2021/yamagishi21_asvspoof.html

[67] [68] [83] [84] [91] [92] [94] [116] [119] ASVspoof 2021 Evaluation Plan
https://www.asvspoof.org/asvspoof2021/asvspoof2021_evaluation_plan.pdf

[121] [163] [164] | ASVspoof
http://www.asvspoof.org/index2021.html

[147] [148] [PDF] aasist: audio anti-spoofing using integrated spectro-temporal
https://www.eurecom.fr/publication/6696/download/sec-publi-6696.pdf

[151] AASIST: Audio Anti-Spoofing using Integrated Spectro-Temporal ...
https://arxiv.org/abs/2110.01200

[158] ASVspoof 2019: Spoofing Countermeasures for the Detection of ...
https://www.researchgate.net/publication/349419506_ASVspoof_2019_Spoofing_Countermeasures_for_the_Detection_of_Synthesized_Converted_and_Replayed_Speech

[161] Official PyTorch implementation of "AASIST - GitHub
https://github.com/clovaai/aasist