December 2020

Automatic Delivery of Machine Learning Models to User Device to Enable App Features

Keun Soo Yim

Neha Pattan

Automatic Delivery of Machine Learning Models to User Device to Enable App Features

ABSTRACT

Many mobile apps, e.g., virtual assistants, navigation apps, video apps, etc., use various machine learning (ML) models. Different features of the app may have respective associated ML models which may often not be available locally on a user device and need explicit user action to download from a server. This disclosure describes techniques for automatic synchronization of machine learning models to a user device. With user permission, ML model(s) of an app that are not available locally or for which a new version is available are automatically downloaded when certain conditions such as are met. The conditions can include network conditions and other device-specific parameters. The overheads for determining whether the conditions are satisfied are reduced by utilizing an on-device cache and flag, which can reduce the impact on app startup latency.

KEYWORDS

- On-device machine learning
- ML model download
- ML model update
- Mobile app update
- Data synchronization

BACKGROUND

Many mobile apps, e.g., virtual assistants, navigation apps, video apps, etc., use various machine learning (ML) models. Different features of the app may have respective associated ML models. When a new feature is launched, it may be made available to a subset of users, e.g., to

user devices within a geographic region, user devices that meet certain criteria, etc. The corresponding ML models then need to be downloaded from the server onto such user devices.

Conventionally, model download is performed (with user permission) when certain conditions are fulfilled, e.g., the device is in charged state and has WiFi connectivity (as opposed to cellular data connection, which may be metered). Thus, it can take substantial time before a new ML model and the corresponding feature is available on a user device. Further, if the download fails for any reason (e.g., poor network connectivity), the user may need to go through the same steps again.

DESCRIPTION

This disclosure describes techniques for automatic synchronization of machine learning models between a server and user device. The techniques are implemented with specific user permission. Per the techniques, it is first determined whether certain predefined conditions are satisfied. Such conditions may include, e.g., whether a flag corresponding to a particular feature is on and whether this is a first time execution of the feature?; determine that the model size is smaller than a threshold (optional); confirm that the device is connected to a network, and that the device is not in roaming condition and has not enabled a data saver setting. If the conditions are met, the ML model corresponding to the feature is downloaded, e.g., by executing a background thread in synchronous or asynchronous mode. The feature is made visible to the user as soon as the model download is complete.

Fig. 1 illustrates an example method for downloading an ML model that is associated with an application feature in mobile devices. User permission is first obtained to enable automatic download of ML models to support app features. Users are provided with options to limit or deny such permission, or to disable automatic downloads entirely.

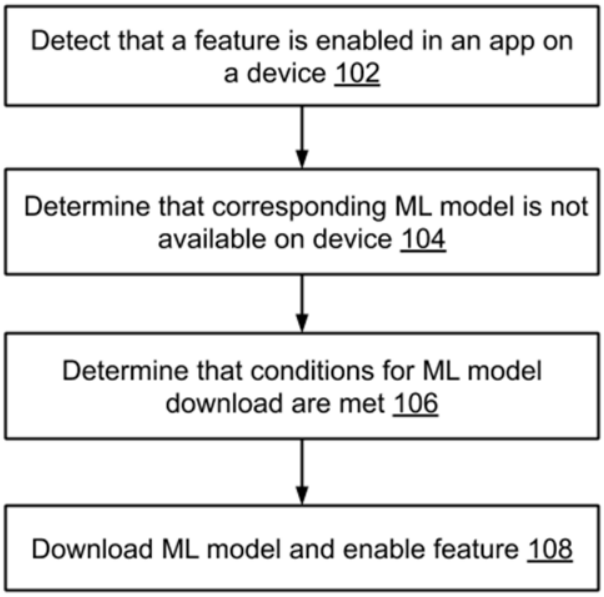

Fig. 1: Example method to selectively download ML models and enable app features

If the user provides the permission for automatic downloads, it is first determined whether a particular feature of an app on a mobile device has been enabled (102). For each enabled feature, it is determined whether a local ML model that supports the feature exists and is current. If the model is not available locally or if the locally-stored model version is not current), it is determined that the ML model for the feature is not available locally and is to be downloaded (104). Prior to initiating model download, it is determined that conditions for model download are met, e.g., that the device has sufficient charge and is connected to a WiFi network (106). When the conditions are met, the ML model is automatically downloaded and the corresponding feature is enabled (108). If there are additional features and/or apps, the process can be repeated to download additional models or model updates.

An important challenge in this process is to optimize the checking overheads (to determine whether the conditions for model download are met) since such checking can negatively impact the startup latency for the app, which is an important metric of app

performance, For example, checking the status of a network connection can be time-consuming and can add to startup latency. The checking overheads can be optimized in multiple ways.

One technique to optimize is to use a usage history flag. With user permission, a usage history flag (that enables a check for first time usage) is stored on disk. The flag is cached in memory and is access-protected using a lock. When the app starts, it is determined if the cache has content associated with usage history and if such content is available, it is retrieved from the cache. If the content is not available in the cache, the value is read from the disk. If the ML model is not available locally and if conditions for model download are fulfilled, the model is downloaded automatically and upon successful download the flag is updated. In case of download failure, the flag is reset.

When a network condition check for model download fails, the usage history flag is reset. Thus, instead of rechecking the network status (which can be time-consuming), the usage history flag is checked. Since the flag is not set, the model is downloaded automatically at a later time when the network condition requirements are fulfilled, e.g., the device is connected to WiFi.

The described techniques enable easy and automatic download of ML models, with automatic enablement of corresponding features without specific action by the user for each such event. The user can enable automatic ML model downloads when certain conditions are fulfilled, e.g., when the device is connected to a WiFi or other free network. The techniques can be used to update any type of app on a mobile device such as a smartphone, tablet, wearable device, laptop computer, etc.

Further to the descriptions above, a user is provided with controls allowing the user to make an election as to both if and when systems, programs, or features described herein may enable collection of user information (e.g., information from sensors of a user device, a user's

current location), and if the user is sent content or communications from a server (e.g., risk map). In addition, certain data are treated in one or more ways before it is stored or used, so that personally identifiable information is removed. For example, a user's identity is treated so that no personally identifiable information can be determined for the user. Thus, the user has control over what information is collected about the user, how that information is used, and what information is provided to the user.

CONCLUSION

This disclosure describes techniques for automatic synchronization of machine learning models to a user device. With user permission, ML model(s) of an app that are not available locally or for which a new version is available are automatically downloaded when certain conditions such as are met. The conditions can include network conditions and other device-specific parameters. The overheads for determining whether the conditions are satisfied are reduced by utilizing an on-device cache and flag, which can reduce the impact on app startup latency.

REFERENCES

1. Gracioli, Giovani, Ahmed Alhammad, Renato Mancuso, Antônio Augusto Fröhlich, and Rodolfo Pellizzoni. "A survey on cache management mechanisms for real-time embedded systems." *ACM Computing Surveys (CSUR)* 48, no. 2 (2015): 1-36.