Detección Inteligente de Sucesos en Smart Cities con Feedback de los Ciudadanos

José M. Aragón-Jurado[1], Luis E. Acuña-Vega[2], Guadalupe Ortiz[1], Juan Boubeta-Puig[1] y Andrés Muñoz[1]

[1] Departamento de Ingeniería Informática, Escuela Superior de Ingeniería, Universidad de Cádiz, España,
{josemiguel.aragon, guadalupe.ortiz, juan.boubeta, andres.munoz}@uca.es
[2] Máster de Investigación en Ingeniería de Sistemas y Computación, Escuela Superior de Ingeniería, Universidad de Cádiz, España,
luis.acunavega@alum.uca.es

Resumen Día a día en las ciudades ocurren sucesos que pueden afectar a la vida de los ciudadanos así como causar costes a las diferentes actividades y servicios que las administraciones ofrecen. La lluvias torrenciales y los incendios son dos de este tipo de sucesos cuya predicción es importante, tanto para salvar vidas como para abaratar costes. En la actualidad, las soluciones se centran en detectar uno de los tipos de sucesos aplicando modelos de aprendizaje computacional, sin tener en cuenta el *feedback* de los ciudadanos ni empleando el procesamiento de eventos complejos. En este artículo proponemos y desarrollamos una arquitectura para la detección inteligente de incendios y lluvias torrenciales en *smart cities* empleando el procesamiento de eventos complejos y, en el caso de los incendios, combinándolo con un modelo de aprendizaje profundo. Además, los mismos ciudadanos pueden participar en la arquitectura identificando si esos sucesos están sucediendo realmente o han sido un falsos positivos, permitiendo el reentrenamiento del modelo. La participación de los ciudadanos en la arquitectura es importante para que los modelos de aprendizaje se puedan adaptar a los cambios que ocurran por la ciudad a lo largo del tiempo, mejorando la calidad de vida.

Keywords: *Smart Cities*, Aprendizaje profundo, Procesamiento de eventos complejos, Detección inteligente, *Feedback*, Aprendizaje supervisado

1 Introducción

En la actualidad, el auge del éxodo rural moderno está provocando la concentración de un gran número de personas en las grandes ciudades. Las ciudades tradicionales no se encuentran preparadas para proveer de servicios de calidad a tal concentración de personas. La consolidación del campo del Internet de las Cosas (IoT) y de la computación en la nube aporta los mecanismos necesarios para poder empezar a resolver esta necesidad. Las *smart cities* tratan de resolver las necesidades de los ciudadanos a través de servicios inteligentes, mejorando la calidad de vida.

Detectar sucesos tales como incendios o lluvia en tiempo real es de gran utilidad a la hora de salvar la vida de las personas o de abaratar costes de las infraestructuras que vayan a ser perjudicadas. No obstante, los modelos predictivos no son perfectos y, en algunas situaciones, pueden provocar mayores costes en lugar de disminuirlos. Consecuentemente, proveer de un sistema de detección de este tipo de sucesos preciso no es una tarea trivial.

Las posibilidades que nos ofrecen las ciudades inteligentes son ilimitadas y éstas suelen mejorar la calidad de vida de los habitantes de la ciudad [8], por lo que la detección de eventos complejos como incendios o la predicción de precipitaciones en el corto plazo utilizando herramientas tecnológicas, está cada día más cerca de ser parte de esta transformación. De forma prioritaria, tener una alarma temprana de potenciales emergencias como los incendios, se hace especialmente importante cuando estos han aumentado en su frecuencia en algunas partes del mundo [9].

Para resolver los problemas y carencias descritas, proponemos una arquitectura de detección de lluvias e incendios en tiempo real para *smart cities*. Además, la arquitectura es retroalimentada con el *feedback* de los ciudadanos que disponen de una aplicación web. A través de la aplicación web, los ciudadanos pueden decir si el suceso realmente está ocurriendo. Para lograr este sistema de detección se integra el procesamiento de eventos complejos (CEP) [10] con modelos de aprendizaje computacional como son las redes neuronales.

El resto del documento se organiza de la siguiente manera: en la Sección 2, se comentan los conceptos teóricos previos necesarios para la realización del trabajo. En la Sección 3, se expone y define la propuesta del trabajo así como la implementación de la arquitectura. Posteriormente, en la Sección 4, se comentan diferentes trabajos relacionados realizados en el mismo ámbito. Finalmente, en la Sección 5, se presentan las conclusiones generales de la arquitectura propuesta y el trabajo futuro.

2 Antecedentes

En esta sección se revisan los conceptos básicos de CEP, bus de servicios empresariales (ESB), interfaz de programación de aplicaciones (API) REST y redes convolucionales utilizados en nuestra propuesta.

2.1 Procesamiento de eventos complejos

CEP es un conjuntos de técnicas y conceptos que permiten el análisis en tiempo real de eventos, que suceden en un entorno dado así como la extracción y generación de conclusiones para identificar posibles amenazas u oportunidades asociadas a los mismos eventos. Los patrones para identificar eventos se definen en un lenguaje específico denominado lenguaje de procesamiento de eventos (EPL).

A lo largo del trabajo se ha empleado como motor de eventos complejos Esper [1], ampliamente conocido y escrito en Java, lo que permite su integración rápida con múltiples aplicaciones.

2.2 Bus de servicios empresariales

Un ESB es una aplicación software que permite la comunicación entre servicios sirviendo como intermediario e integrador [13]. Además, provee de una capa de abstracción para esta comunicación. Permite emplear múltiples protocolos, tiene un diseño modular y hace uso de la concurrencia. Como su nombre lo indica, suele estar orientado a servicios usando este patrón de diseño.

Mule [2] es una conocida implementación de ESBs que permite la integración de multitud de plataformas en la nube así como herramientas y servicios.

2.3 API REST

Una API es el intermediario que permite la interacción entre un usuario y un servidor o recurso, permitiendo desacoplar el servicio del consumidor y proveyendo funcionalidades como seguridad, alta disponibilidad, entre otras. No obstante, una API REST es aquella que aplica los principios de la transferencia de estado representacional (REST). Este tipo de APIs usan métodos del protocolo HTTP como son POST, GET, PUT y DELETE.

En las APIs REST, el servidor no guarda el estado del cliente, por lo que, cada llamada es independiente de las demás. De esta manera se establece una interfaz uniforme entre cliente y servidor donde la información se transfiere de forma estandarizada.

2.4 Redes neuronales convolucionales

Las redes neuronales se tratan de un modelo computacional compuesto por una serie de neuronas artificiales o perceptrones. Los perceptrones están formados por una serie de conexiones de entrada, una función de activación y una serie de conexiones de salida. Cada conexión de entrada lleva asociado un peso para ponderar cada uno de los valores de entrada. Consecuentemente, dentro de la neurona, se computa una regresión lineal con un peso adicional conocido como bias o sesgo. Al resultado de la regresión se le aplica la función de activación, computándose así, la salida de la neurona. Las neuronas se agrupan en capas que envían sus resultados a la siguiente capa como entrada hasta llegar a la capa de salida.

Las redes neuronales convolucionales son un tipo especial de redes neuronales donde las neuronas se encargan de detectar diferentes propiedades en imágenes, siendo ampliamente usadas para tareas de reconocimiento de imágenes. Se encuentran compuestas principalmente por capas convolucionales que aplican la operación de la convolución con un filtro a las distintos píxeles de la imagen, capas reductoras que reducen las dimensiones de la imagen, y, capas de clasificación, o redes neuronales clásicas.

3 Arquitectura propuesta para la detección inteligente de sucesos

En este sección se procede a describir cómo hemos diseñado la arquitectura propuesta, integrando las diferentes tecnologías. Se trata de una novedosa arquitectura que trata de detectar tanto incendios como lluvias en tiempo real para avisar a los distintos ciudadanos. La novedad o innovación radica en el uso del *feedback* de los clientes para retroalimentar y mejorar nuestro sistema.

3.1 Arquitectura

La integración de las diferentes tecnologías para formar nuestra arquitectura propuesta para la detección inteligente de sucesos se muestra en la Figura 1. Los productores de los diferentes eventos simples son los diferentes sensores repartidos por la ciudad, describiéndose en la Sección 3.2 cada uno de los datos sensorizados. A partir de estos eventos se produce un filtrado generando eventos complejos para fuego y lluvia en tiempo real de acuerdo a unos patrones previamente definidos. Para los eventos complejos sobre incendios detectados, una red neuronal convolucional se ha empleado para detectar si existe realmente un fuego, cuyo resultado será enviado a través de una API REST a los ciudadanos. Para ello, activará las cámaras situadas en la zona donde se encuentran los sensores, empleando dicha imagen para detectar si hay un incendio realmente. Consecuentemente, los ciudadanos son los consumidores de eventos de nuestra arquitectura, que disponen de una aplicación web con un mapa de la ciudad desde donde se les notificará del momento y lugar de incendios y lluvias. El ciudadano podrá verificar si realmente se está produciendo el evento y notificar si la predicción es correcta o errónea, almacenándose dicha información en una base de datos para que al final del día se emplee para reentrenar al modelo predictivo.

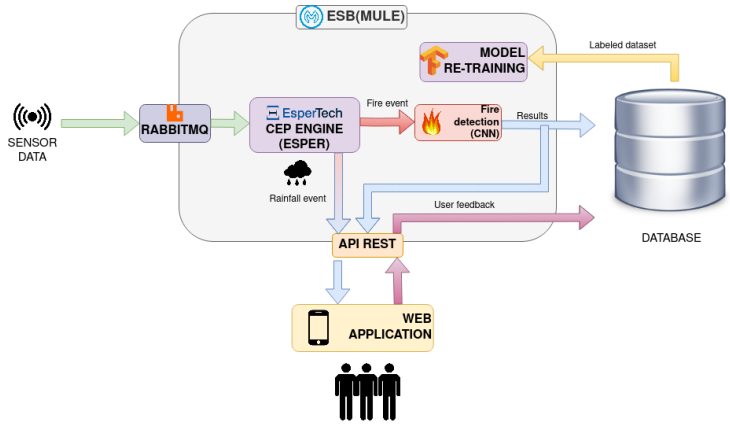

Fig. 1: Arquitectura propuesta para la detección inteligente de sucesos

A continuación se explican cada una de las tecnologías integradas en nuestra arquitectura así como su funcionalidad dentro de la misma:

- **Sensores físicos**. Encargados de medir diferentes parámetros en la *smart city* generando un evento simple por cada medición realizada.
- **Bróker de mensajería**. Recibe los eventos simples a través del protocolo AMQP y los envía al motor CEP, utilizando el software de RabbitMQ como bróker.
- **Motor CEP**. Sobre él se definen los diferentes patrones de eventos complejos. Recibe los distintos eventos simples y genera los eventos complejos que cumplan los patrones, empleándose la implementación de Esper.
- **Base de datos**. Se almacenan los distintos eventos complejos tanto de lluvias torrenciales como de incendios junto con el *feedback* asociado aportado por los usuarios. Como sistema de gestión de base datos se ha empleado MariaDB.
- **Red neuronal convolucional**. Recibe los eventos complejos de incendio, activando las cámaras de la zona. A partir de la imagen de la zona recién tomada, realiza una predicción de si realmente está ocurriendo un incendio.
- **Aplicación web**. Muestra un mapa de la ciudad junto con los diferentes sucesos detectados. Los ciudadanos pueden acceder a ella desde el teléfono móvil e indicar si los sucesos están ocurriendo realmente o no, retroalimentando al sistema. React ha sido el framework sobre el que se ha implementado la aplicación.
- **API REST**. Sirve de interfaz para enviar los eventos complejos a la aplicación web así como el *feedback* de los usuarios a la base de datos.

La tecnología empleada para integrar fácilmente cada una de las tecnologías mencionadas previamente es el ESB, concretamente su implementación Mule.

3.2 Datos sensorizados

Los diferentes tipos de datos recogidos por los sensores se agrupan por zonas y son: la temperatura actual en ^{o}C, el porcentaje de nubosidad, la presión en hectopascales, la precipitación en milímetros de agua y, la intensidad luminosa en candelas.

Por otro lado, cada vez que se produce un evento complejo del tipo incendio se realiza una fotografía de la zona donde los sensores se encuentran situados, usándola en la clasificación de nuestra red convolucional detectando así si realmente hay un incendio en la zona.

3.3 Patrones de eventos

Los patrones de eventos para la implementación del primer filtro en la arquitectura se definieron según las siguientes restricciones. En la Figura 2 el área coloreada representa las combinaciones de temperatura y humo que definirán la generación del evento de candidato a incendio.

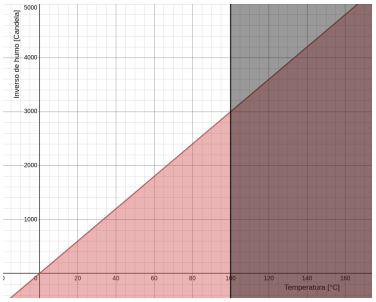

Fig. 2: Umbrales para el filtrado de eventos complejos de incendio

Para las precipitaciones se dividió las variables en los intervalos: presión creciente y no creciente, temperatura alta y baja, nubosidad y precipitaciones previas como alta, media y baja. La Tabla 1 representa las veces necesarias que se ha detectado algún evento simple para que se de un escenario para generar un evento complejo. Por ejemplo, en el caso de presión creciente, temperatura alta y nubosidad media, se necesitan detectar tres valores de precipitación media para detectar una lluvia torrencial. A la hora de definir los diferentes esquemas

Tabla 1: Frecuencias requeridas para detectar un evento de precipitación

Presion+/Temperatura+	Precipitación Alta	Precipitación media	Precipitación baja
Nubosidad Alta	1	1	1
Nubosidad Media	1	3	NO
Nubosidad Baja	1	NO	NO
Presión+/Temperatura-	Precipitación Alta	Precipitación media	Precipitación baja
Nubosidad Alta	1	1	1
Nubosidad Media	1	3	NO
Nubosidad Baja	1	5	NO
Presión-/Temperatura+	Precipitación Alta	Precipitación media	Precipitación baja
Nubosidad Alta	1	1	1
Nubosidad Media	1	1	1
Nubosidad Baja	1	5	NO
Presión+/Temperatura-	Precipitación Alta	Precipitación media	Precipitación baja
Nubosidad Alta	1	1	1
Nubosidad Media	1	1	1
Nubosidad Baja	1	3	5

para los eventos simples en Esper, almacenamos el tipo de evento simple, el valor medido, la latitud y la longitud del sensor para cada uno de los datos especificados en la Sección 3.2.

Finalmente se definen dos patrones adicionales que generan eventos complejos de incendio y de precipitación, empleando este último un evento complejo

adicional que calcula la presión atmosférica media en un minuto. El patrón para detectar eventos complejos de tipo incendio es descrito a continuación:

```
@public @buseventtype create json schema Fire as (candela int, celsius int,
latitude int, longitude int)
@Name("Fire")
select e1.latitude as latitude, e1.longitude as longitude, e1.value as celsius,
e2.value as candela
from pattern [every (
    (e1 = temperature(e1.value > 100))
    or
    (e1 = temperature()-> e2 = smoke(e2.value <= (30*e1.value)
    and latitude = e1.latitude and longitude = e1.longitude)))]
```

Como observamos en la Tabla 1, el código Esper EPL del patrón complejo de precipitaciones tiene 30 escenarios de precipitaciones por lo que por su extensión es demasiado amplia para estar presente en el documento.

3.4 Flujos

El flujo inicial de la aplicación comienza con los datos recibidos de los sensores a través de RabbitMQ, empleando el protocolo AMQP. Los eventos simples recibidos son procesados por Esper.

En el flujo para la detección de eventos complejos, estos últimos se almacenan en sus respectivas tablas de la base de datos y, en el caso de incendio o lluvia, dichos eventos se envían a través de una API REST a la aplicación web. Además, se realiza una llamada a la red neuronal convolucional cuando se detecta un incendio, llamando así al modelo clasificador y detectar si realmente hay un incendio.

Con respecto al flujo para almacenar el *feedback* enviado por los ciudadanos, se define una API REST a partir de la cual recibimos las respuestas de los ciudadanos, almacenándolas en sus respectivas tablas de la base datos. En el caso de los incendios, los datos almacenados se emplearán en el reentrenamiento de la red neuronal.

Un último flujo se activa cada vez que detecta un fichero en un directorio previamente especificado, con el objetivo de realizar el reentrenamiento de la red neuronal convolucional con el *feedback* almacenado de los ciudadanos. De esta forma, cada cierto tiempo, se reentrenará el modelo de aprendizaje, adaptándose a los cambios en el entorno.

3.5 Aprendizaje y retroalimentación del modelo predictivo

En relación a la red neuronal convolucional se propone una arquitectura basada en el modelo InceptionV3 [15], empleando los pesos ajustados para el conjunto de datos Imagenet [5] y añadiéndole al modelo una capa reductora seguida de dos capas básicas con la función de activación *Rectified Lineal Unit* (RELU) y una capa final con dos neuronas la cual realiza la clasificación. Las capas añadidas al modelo deben ser previamente entrenadas en un conjunto de imágenes etiquetadas con el objetivo de realizar la predicción de los incendios.

A la hora del reentrenamiento del modelo, cada día los incendios predichos incorrectamente almacenados en la base de datos se emplean para ajustar los diferentes pesos de la red neuronal. Para ello, nuestra metodología se basa en emplear tanto con el conjunto de datos etiquetados como el nuevo para entrenar la red con un ratio de aprendizaje muy pequeño, evitando así que el modelo olvide todo lo aprendido.

4 Trabajos Relacionados

En la literatura se han propuesto trabajos sobre la predicción de incendios en *smart cities* [14] empleando técnicas basadas en el procesamiento de imágenes. Sin embargo, la retroalimentación con la información de los ciudadanos no es tenida en cuenta en su modelo. En [12] se realiza un estudio general sobre los modelos basados en redes neuronales enfocados en la predicción de lluvias mientras que en [3] se propone un modelo concreto. Aún así, ninguno de estos modelos se usa para la detección de lluvias a un corto plazo. Por último, un esquema de aprendizaje semisupervisado es propuesto en [11] para aprovechar el *feedback* de los ciudadanos, no obstante, no existe su aplicación a la predicción de fenómenos como incendios o lluvias torrenciales en *smart cities*.

Existen variadas arquitecturas que intentan resolver el problema de detección temprana de incendios, tal como se ve en [14] el cual utiliza un novedoso sistema de drones con cámaras y sensores para determinar la aparición de un incendio. En [16] se utiliza una simple arquitectura asociada a una cámara y un sistema de análisis de detección de llamas usando un algoritmo de mezclas gaussianas, al igual que en [7], aunque usando redes neuronales convolucionales y una transformada *wavelet*. A su vez en [4] se aproxima al problema con una red convolucional de región y análisis de textura espacial permitiendo un aumento en los verdaderos positivos y reduciendo los falsos positivos con respecto al estado del arte. No obstante, ninguno de estos trabajos se combinan con la tecnología CEP en el ámbito de las *smart cities* ni definen una estrategia de reentrenamiento del modelo para adaptarse a los cambios en el medio.

El problema de predicción de precipitaciones en el corto plazo ha sido tratado en [17], donde se señala que las actuales soluciones de predicción de corto plazo obtienen baja precisión, creando un nuevo algoritmo *Regional Combined short-term rainfall Forecasting approach* (DRCF) que llegó a obtener mejores resultados que los modelos propuestos en la literatura. Otro ejemplo asociado a las precipitaciones se puede ver en [6] donde se propone una arquitectura para la detección de sectores de acumulación de agua que generan congestión, basándose en la altura geo-referenciada de la ciudad en combinación con la utilización sensores de precipitación, pudiendo ser capaz de estimar si habrá una inundación en un lugar determinado. Sin embargo, el uso de CEP para generar alarmas de posibles precipitaciones a corto plazo no ha sido estudiado, pudiendo aprovechar su uso para la detección rápida de precipitaciones gracias a su bajo coste respecto a los modelos de aprendizaje.

Como hemos visto existen multitud de sistemas para la detección de incendios y modelos de predicción lluvias en corto plazo, no obstante, nuestra arquitectura integra la detección de ambos sucesos en un mismo sistema junto con el uso de la tecnología CEP, empleando su rápida gestión de eventos. Además, los ciudadanos de las *smart cities* participan e influyen en el funcionamiento del modelo gracias all empleo de su *feedback* en el reentrenamiento de la red neuronal convolucional.

5 Conclusiones y Trabajo Futuro

En este trabajo se ha presentado una arquitectura software para la detección de sucesos como incendios o lluvias en el ámbito de las *smart cities*, empleando una red neuronal convolucional y la detección de eventos complejos. Por otro lado, los ciudadanos influyen activamente en el funcionamiento y desarrollo de la arquitectura debido a que su *feedback* sobre los distintos eventos detectados es almacenado, empleándose posteriormente para el reajuste del modelo de detección de incendios.

Gracias al empleo de un ESB, hemos conseguido integrar multitud de componentes diferentes que nos permite utilizar CEP como filtrado de entradas a clasificar por la red neuronal, diferenciándonos del resto de trabajos en el ámbito. Asimismo, el reentrenamiento del modelo predictivo otorga flexibilidad a nuestro sistema, pudiendo adaptarse a cambios en el entorno gracias a la participación activa de los ciudadanos.

Como líneas de trabajo futuro, planeamos realizar un estudio cuantitativo de cómo afecta el feedback aportado por los ciudadanos a nuestro modelo de detección, analizando la ganancia obtenida. En base a los resultados del estudio, se integrará un mecanismo para suavizar posibles ataques malintencionados empleando la retroalimentación de los usuarios. Asimismo, se tratará de integrar un modelo de aprendizaje para predecir lluvias a corto plazo junto con un esquema de patrones de eventos complejos adaptables en función del feedback recibido. Además, se realizará una evaluación de diferentes estrategias de reentramiento de los modelos de aprendizaje empleados con el objetivo de definir la metodología que mejor se ajuste a nuestro sistema.

Otro trabajo a realizar en el futuro es la implementación y monitorización de un caso de uso real en una ciudad para comprobar su efectividad, analizando la participación de los ciudadanos. Finalmente, los resultados podrían ser aplicables a la integración de otros dominios y verticales de ciudades inteligentes, tales como podrían ser la monitorización y toma de decisiones inteligentes en redes de abastecimiento de agua y su correlación con la monitorización de la vida asistida por el entorno.

Agradecimientos

Este trabajo ha sido soportado parcialmente por los proyectos ALLEGRO (PID2020-112827GB-I00), financiado por MCIN/AEI/10.13039/501100011033, y DECISION (P20_00865), financiado por Plan Andaluz de Investigación, Desarrollo e

Innovación (PAIDI 2020), cofinanciado en un 80% por la Unión Europea, en el marco del Programa Operativo FEDER Andalucía 2014-2020 "Crecimiento inteligente: una economía basada en el conocimiento y la innovación".

Referencias

1. EsperTech: Esper - Complex Event Processing, https://www.espertech.com/esper/
2. MuleSoft: Mule ESB, https://developer.mulesoft.com/
3. Abhishek, K., Kumar, A., Ranjan, R., Kumar, S.: A rainfall prediction model using artificial neural network. In: 2012 IEEE Control and System Graduate Research Colloquium. pp. 82–87. IEEE (2012)
4. Barmpoutis, P., Dimitropoulos, K., Kaza, K., Grammalidis, N.: Fire detection from images using faster r-cnn and multidimensional texture analysis. In: ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 8301–8305 (2019)
5. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
6. Gupta, A., Bansal, A., Gupta, R., Naryani, D., Sood, A.: Urban waterlogging detection and severity prediction using artificial neural networks. In: 2017 IEEE 19th International Conference on High Performance Computing and Communications; IEEE 15th International Conference on Smart City; IEEE 3rd International Conference on Data Science and Systems (HPCC/SmartCity/DSS). pp. 42–49 (2017)
7. Huang, L., Liu, G., Wang, Y., Yuan, H., Chen, T.: Fire detection in video surveillances using convolutional neural networks and wavelet transform. Engineering Applications of Artificial Intelligence 110, 104737 (2022)
8. Islam, T., Mukhopadhyay, S., Suryadevara, N.: Smart sensors and internet of things: A postgraduate paper. IEEE Sensors Journal 17(3), 577–584 (2016)
9. Jenner, L.: Forest fire numbers rise in india in 2016, fire and smoke. National Aeronautics and Space Administration (NASA) (2017)
10. Luckham, D.: The power of events, vol. 204. Addison-Wesley Reading (2002)
11. Mohammadi, M., Al-Fuqaha, A., Guizani, M., Oh, J.S.: Semisupervised deep reinforcement learning in support of IoT and smart city services. IEEE Internet of Things Journal 5(2), 624–635 (2017)
12. Nayak, D.R., Mahapatra, A., Mishra, P.: A survey on rainfall prediction using artificial neural network. International Journal of Computer Applications 72(16) (2013)
13. Rademakers, T., Dirksen, J.: Open-Source ESBs in Action: Example Implementations in Mule and ServiceMix. Simon and Schuster (2008)
14. Sharma, A., Singh, P.K., Kumar, Y.: An integrated fire detection system using IoT and image processing technique for smart cities. Sustainable Cities and Society 61, 102332 (2020)
15. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1–9 (2015)
16. Töreyin, B.U., Dedeoğlu, Y., Güdükbay, U., Çetin, A.E.: Computer vision based method for real-time fire and flame detection. Pattern Recognition Letters 27(1), 49–58 (2006)
17. Zhang, P., Jia, Y., Gao, J., Song, W., Leung, H.: Short-term rainfall forecasting using multi-layer perceptron. IEEE Transactions on Big Data 6(1), 93–106 (2020)